Burroughs

Language Manual

B 1000 Systems

SORT

(Relative to Mark 11.0 Release)

Priced Item
Printed in U.S.A.
September 1983

1152071

Burroughs

Language Manual

B 1000 Systems SORT

(Relative to Mark 11.0 Release)

Copyright © 1983, Burroughs Corporation, Detroit, Michigan 48232

Priced Item
Printed in U.S.A.
September 1983

1152071

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded using the Remarks form at the back of the manual, or may be addressed directly to Corporate Documentation-West, Burroughs Corporation, 1300 John Reed Court, City of Industry, California 91745, U.S.A.

LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii	Original
iii	Original
iv	Blank
v thru vii	Original
viii	Blank
1-1 thru 1-7	Original
1-8	Blank
2-1 thru 2-25	Original
2-26	Blank
3-1 thru 3-29	Original
3-30	Blank
A-1 thru A-10	Original
Index-1 thru Index-2	Original

TABLE OF CONTENTS

Section	Title	Page
1	INTRODUCTION	1-1
	General	1-1
	Document Overview	1-1
	SORT Specification Programs	1-1
	SORT Intrinsic Programs	1-1
	Sorting in COBOL	1-2
	Syntax Diagrams	1-2
	Required Items	1-3
	Optional Items	1-3
	Loops	1-4
	Bridges	1-4
	File Naming Conventions	1-5
	File Title	1-5
	File Name	1-5
	File Identifier	1-6
	Related Documents	1-7
2	SORT	2-1
	SORT Program Execution	2-1
	Card Reader Input	2-1
	ODT Keyboard Input	2-1
	Comments	2-1
	Compiling Sort Programs	2-2
	Listing	2-2
	SORT Specifications	2-2
	FILE Specification	2-2
	Input File Description	2-3
	Input File Description Using File Attributes	2-5
	Output File Description	2-7
	Output File Description Using File Attributes	2-9
	KEY Specification	2-11
	Option Specifications	2-13
	BIAS Option	2-14
	COLLATE Option	2-14
	DISKSORT Option	2-15
	DUPCHECK Option	2-16
	INCLUDE/DELETE Option	2-17
	INPLACE Option	2-18
	MEMORY Option	2-18
	MERGE Option	2-19
	NOPRINT Option	2-19
	OVERRIDE Option	2-19
	PARITY DISCARD Option	2-19
	PARTITION Option	2-20
	RECORDS Option	2-20
	RESTART Option	2-20

TABLE OF CONTENTS (Cont)

Section	Title	Page
	SEQUENCE Option	2-21
	STABLE Option	2-21
	SYNTAX Option	2-21
	TAGCOBOL Option	2-21
	TAGRPG Option	2-22
	TAGSEARCH Option	2-22
	TAGSORT Option	2-22
	TAPESORT Option	2-22
	TIME Option	2-23
	TIMING Option	2-23
	WORKPACK Option	2-24
	ZIP Option	2-24
	SORT Reserved Words and Characters	2-25
3	SORT/UTILITY	3-1
	SORT/UTILITY Program Execution	3-1
	Card Reader Input	3-1
	ODT Keyboard Input	3-2
	Comments	3-2
	Compiling Sort Programs	3-2
	Free-Form Format Specifications	3-2
	FILE Specification	3-3
	Input File Description	3-3
	Input File Description Using File Attributes	3-5
	Output File Description	3-7
	Output File Description Using File Attributes	3-8
	Option Specifications	3-10
	COLLATE Option	3-11
	CONTROL Option	3-12
	DROP CONTROL Option	3-12
	FIXED Option	3-12
	MEMORY Option	3-12
	NOPRINT Option	3-13
	NOSORT Option	3-13
	RESTART Option	3-13
	STABLE Option	3-13
	SUMMARIZE Option	3-14
	SYNTAX Option	3-14
	TAGSORT Option	3-14
	TEACH Option	3-14
	TIMING Option	3-15
	ZIP Option	3-15
	Record Specifications	3-16
	INCLUDE/DELETE Specification	3-16
	KEY Specification	3-18
	FORCEKEY Specification	3-19
	DATA Specification	3-20

TABLE OF CONTENTS (Cont)

Section	Title	Page
	FORCEDATA Specification	3-21
	SUMMARYDATA Specification	3-21
	OVERFLOW Specification	3-22
	Fixed Format Specifications	3-22
	Header Specification	3-23
	ALTSEQ Specification	3-24
	Include and Delete Specification	3-25
	Field Specification	3-27
	SORT/UTILITY Program Example Using Fixed Format	3-29
A	SORT INTRINSIC TABLES	A-1
	Sort Information Table	A-2
	Sort Key Table	A-6
	Delete Key Table	A-7
	Merge Input Table	A-8
	Sort Communicate	A-9

SECTION 1
INTRODUCTION

General

The B 1000 data processing system sorting and merging facilities consist of two sort specification programs that accept user-supplied sort specifications and four sort intrinsic programs that perform the actual sorting or merging.

Document Overview

A brief description of each section and appendix follows:

Section 1: Introduction
Section 1 provides a brief overview of the sort specification program and the sort intrinsic programs. It also describes the syntax notation and file naming conventions.

Section 2: SORT
Section 2 describes the input specifications to the SORT program.

Section 3: SORT/UTITITY
Section 3 describes the input specifications to the SORT/UTILITY program.

Appendix A: SORT Intrinsic Tables
Appendix A describes the information tables used by the sort intrinsic programs.

SORT Specification Programs

The two sort specification programs are SORT and SORT/UTILITY. Both of these programs accept sort specifications, build tables of parameters, and invoke a sort intrinsic.

The SORT program allows the user to optimize sorting based on system and user requirements. Various types of sorts may be invoked, allowing the user to choose the best sorting method for the particular application.

The SORT/UTILITY program allows sophisticated manipulation of the files during the sort or merge.

SORT Intrinsic Programs

There are four sort intrinsic programs. The SORT/VSORT intrinsic is the usual intrinsic invoked for a sort. The SORT/QSORT (INPLACE), SORT/TAPESORT and SORT/MERGE programs may be invoked depending on the user-supplied specifications to the SORT program.

The SORT/VSORT intrinsic program sorts by a vector replacement method. The SORT/VSORT intrinsic program uses two intermediate work files on disk to build sorted strings of records from the input file. The strings are then merged until the final (output) string is produced. This intrinsic program requires a disk file work space of approximately 2.5 times the input file size, plus the input file size if it is on disk and is not closed with purge.

The SORT/QSORT (INPLACE) intrinsic program reads two blocks of records at a time from the input file, sorts them, and writes them back to the same file on disk. Because the two blocks of records being manipulated are in memory, no sort work space is required on disk. If the output file identifier is different from the input file identifier, the file is copied before sorting.

The SORT/TAPESORT intrinsic program operates similarly to the SORT/VSORT intrinsic program, but the work files are on magnetic tape. Each work tape associated with the work files must be large enough to hold the number of records assigned to it by the SORT/TAPESORT intrinsic program. The number of records in each work file are less than the number of records in the complete file, but the records in the complete file are not evenly distributed to each of the work files.

The SORT/MERGE intrinsic program merges several input files together to create one output file. Each input file is assumed to be already sorted using the same keys. If the input files are not sorted, a warning message is displayed.

Sorting in COBOL

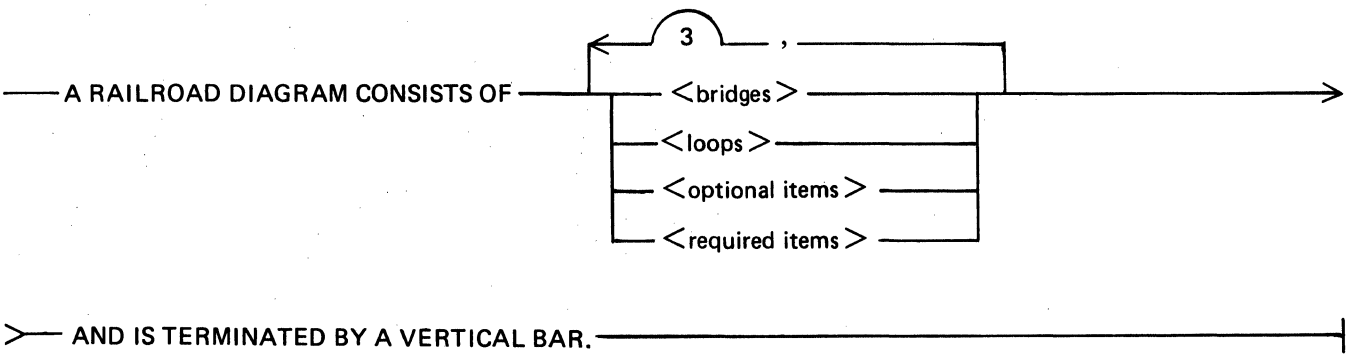
The SORT/VSORT intrinsic program is invoked by using the SORT verb with the sort file assigned to disk in either a COBOL '68 or COBOL '74 program. The SORT/QSORT intrinsic program is invoked by using the SORT verb with the INPLACE option and the sort file assigned to disk in a COBOL '68 program. The SORT/TAPESORT intrinsic program is invoked by using the SORT verb with the sort file assigned to multiple tapes in either a COBOL '68 or COBOL '74 program. The SORT/MERGE intrinsic program is invoked by using the MERGE verb in either a COBOL '68 or COBOL '74 program. The input files are specified by the USING clause, and the output file is specified by the GIVING clause. If the INPUT PROCEDURE and OUTPUT PROCEDURE options of the COBOL SORT verb are used, then an intermediate file is generated and sorted with the file attributes specified in the sort SD specification. Refer to the B 1000 Systems COBOL Reference Manual, form number 1057197, and the B 1000 Systems COBOL74 Reference Manual, form number 1108883, for additional information.

Syntax Diagrams

The syntax of each sort specification to the SORT and SORT/UTILITY program is illustrated by Railroad diagrams. Railroad diagrams show how syntactically valid statements can be constructed. Traversing a railroad diagram from left to right, or in the direction of the arrow heads, and adhering to the limits illustrated by bridges will produce a syntactically valid statement. Continuation from one line of a diagram to another is represented by a right arrow appearing at the end of the current line and at the beginning of the next line. The complete syntax diagram is terminated by a vertical bar |.

Items contained in broken brackets (< >) are syntactic variables which are further defined or which require the user to supply the information requested.

Upper-case items must appear literally. Minimum abbreviations of upper-case items are underlined.



G50051

The following syntactically valid statements may be constructed from the above diagram:

A RAILROAD DIAGRAM CONSISTS OF <bridges> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <optional items> AND IS TERMINATED BY A VERTICAL BAR.

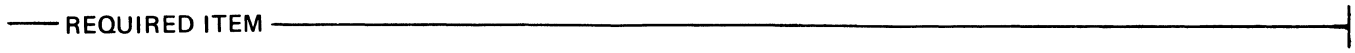
A RAILROAD DIAGRAM CONSISTS OF <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <optional items>, <required items>, <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

Required Items

No alternate path through the railroad diagram exists for required items or required punctuation.

Example:

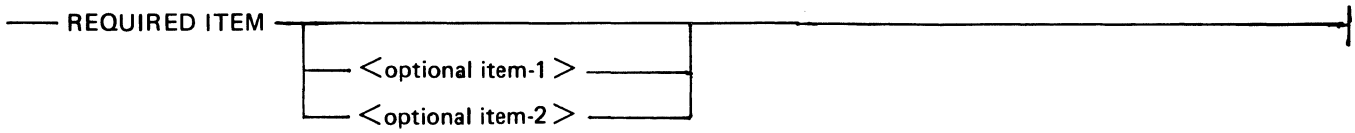


G50052

Optional Items

Items shown as a vertical list indicate that the user must make a choice of the items specified. An empty path through the list allows the optional item to be absent.

Example:



G50053

The following valid statements may be constructed from the above diagram:

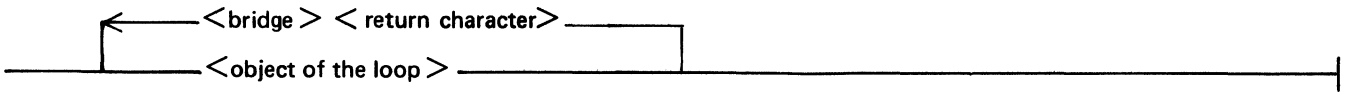
REQUIRED ITEM

REQUIRED ITEM <optional item-1>

REQUIRED ITEM <optional item-2>

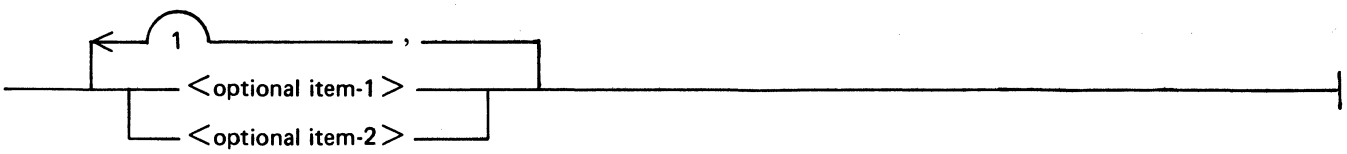
Loops

A loop is a recurrent path through a railroad diagram and has the following general format.



G50054

Example:



G50055

The following statements can be constructed from the railroad diagram in the example.

<optional item-1>

<optional item-1>,<optional item-1>

<optional item-2>,<optional item-1>

A <loop> must be traversed in the direction of the arrow heads, and the limits specified by bridges cannot be exceeded.

Bridges

A bridge illustrates the minimum or maximum number of times a path may be traversed in a railroad diagram.

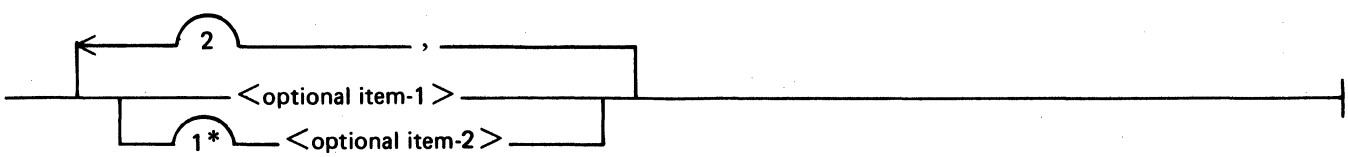
There are two forms of <bridges>.

n is an integer which specifies the maximum number of times the path may be traversed.

n is an integer which specifies the minimum number of times the path must be traversed.

G50056

Example:



G50057

The loop may be traversed a maximum of two times; however, the path for <optional item-2> must be traversed at least one time.

The following statements can be constructed from the railroad diagram in the example.

<optional item-1>,<optional item-2>

<optional item-2>,<optional item-2>,<optional item-1>

<optional item-2>

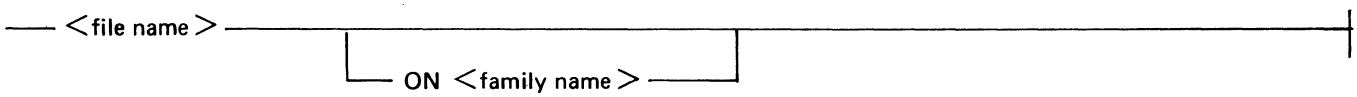
File Naming Conventions

The name of a file is specified by the use of either a file title or a file identifier.

File Title

A <file title> is used to reference a file and the name of the disk on which it resides.

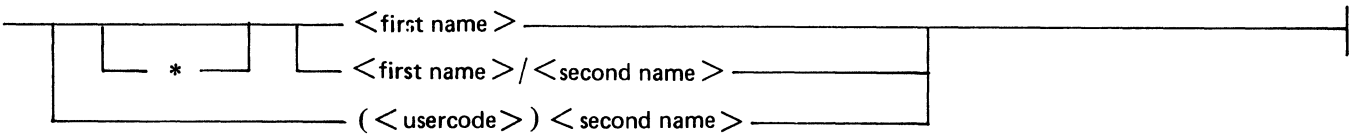
Syntax:



File Name

A <file name> is used to reference a file without regard to the media on which it resides.

Syntax:



Semantics:

<family name>
A <family name> is the name of a disk or a pack. A <family name> consists of one to ten characters. A <family name> of DISK can be used to specify the system disk. DISK will override the default pack associated with a usercode.

*
The asterisk character is used in a <file name> to override the default usercode and the default pack <family name> associated with that usercode. It is only necessary to use an asterisk in a <file name> when the program accessing the file is run under a usercode and the <file name> consists of a <first name> only. Use a <family name> of DISK to override the default pack.

<first name>
A <first name> is the first part of a <file name>. A <first name> consists of one to ten characters. When the asterisk syntax is used, the asterisk character counts as one of the ten characters in the <first name>.

<second name>
A <second name> is the second part of a <file name>. A <second name> consists of one to ten characters.

<usercode>
A <usercode> is a name assigned to a user to secure system and file access. When a <usercode> in parentheses is the first part of a <file name>, the file belongs to the user represented by <usercode>. A <usercode> consists on one to eight characters.

Pragmatics:

The characters, which can be used in a <family name>, <first name>, <second name> or <usercode>, are upper-case letters, digits, underscores and hyphens.

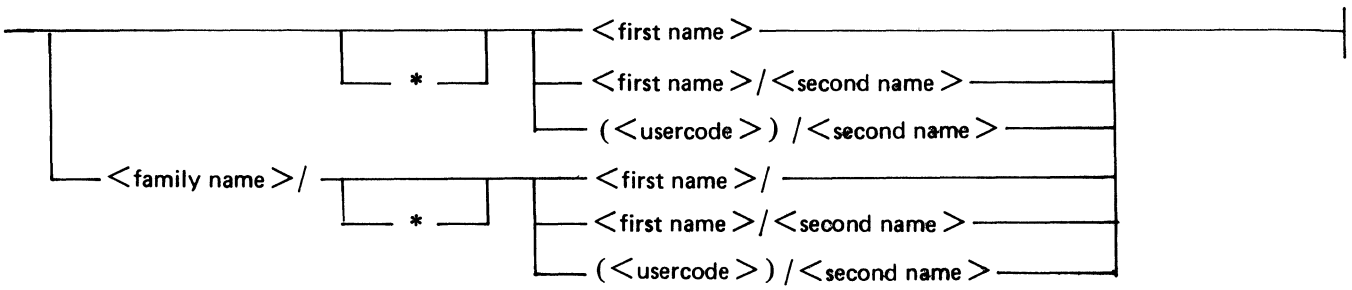
Examples:

```
PARTNO
AA/B
AAA/BBB ON XPACK
(USERA)BBB
(PERS)PAYROLL ON CCC
```

File Identifier

A <file identifier> is used to identify and reference a file.

Syntax:



Semantics:

The elements used in the file identifier syntax have the same meaning as the elements in the file title syntax, with one addition. When the asterisk syntax is used with a <usercode>, the asterisk character counts as one of the eight characters in the <usercode>.

Pragmatics:

The characters, which can be used in a <family name>, <first name>, <second name> or <user-code>, are letters, digits and special characters. When a name is used as part of the SORT or SORT/UTILITY program specifications and the name contains the special characters blank, virgule (/), quotation mark ("), or parentheses, then the name must be enclosed in quotation marks. To specify a quotation mark within a name, a double quotation mark ("") is used. Note that a <family name> is valid only for disk files.

Examples:

```
BBB
BBB/CCC
AAA/BBB/CCC
AAA/BBB/
"A A"/"(USERA)"/"A""/"
```

Related Documents

The following documents are referenced in this manual.

B 1000 Systems COBOL Reference Manual, form number 1057197.

B 1000 Systems COBOL74 Reference Manual, form number 1108883.

B 1000 Systems System Software Operation Guide, Volume 1

SECTION 2
SORT

The SORT program is used to sort or merge data files. The SORT program analyzes user specifications and invokes one of the sort intrinsics to do the sort. The specifications to the SORT program specify the name and hardware media of the input and output files, the location and type of keys, and various sort options.

SORT Program Execution

The procedures used to execute the SORT program are explained in the following paragraphs.

Card Reader Input

The SORT program execution card deck consists of an EXECUTE card, a DATA card, the sort specification cards, and an END card.

Example:

```
? EXECUTE SORT
 ? DATA CARDS
 (specification cards)
 ? END
```

ODT Keyboard Input

If the SORT program is executed with switch 0 or switch 8 set to 1, then the card input file is not used and specifications are entered on the Operator Display Terminal (ODT) as a series of ACCEPT (AX) messages. A blank (null) ACCEPT message terminates the series and begins the sort procedure.

Comments

A colon (:) terminates the scanning of a sort specification record. Any comment may follow a colon. Comments may also be interspersed between sort statements, but must not contain the SORT program reserved words. The colon is ignored as a comment indicator if it appears in a ZIP statement, file identifier, or within quotation marks.

Examples:

```
THIS IS A COMMENT
FILE IN A/RANDOM (DISK DEFAULT)
:RESERVED WORDS MUST FOLLOW A COLON IF IN A COMMENT
FILE OUT A/ORDERED (DISK DEFAULT)
KEY (68 2 A ALPHA) : THIS IS A COMMENT ON THE KEY SPECIFIED
```

Compiling Sort Programs

The SORT program is either executed immediately or compiled to library for subsequent execution. If the SORT program is executed by an EXECUTE control instruction, the specifications are processed and the proper sort intrinsic is immediately invoked provided that no syntax errors are detected. If, however, the SORT program is executed by a COMPILE TO LIBRARY control instruction, an object program is compiled to the disk library for execution at a later time. For example, the following control string compiles an object sort program named SORTPROG to disk:

```
CO[redacted]SORTPROG WITH SORT TO LIBRARY; SW0=1
```

This compiled sort program may be executed at any time thereafter, for example:

```
EXECUTE SORTPROG
```

The four options SEQUENCE, INCLUDE IN, DELETE IN, and DUPCHECK (with parameters) are not allowed when a sort program is compiled to library. DUPCHECK may be specified if the parameters IN and OUT are omitted.

For more information on the EXECUTE or COMPILE control instructions, refer to the B 1000 Systems System Software Operation Guide, Volume 1, form number 1151982.

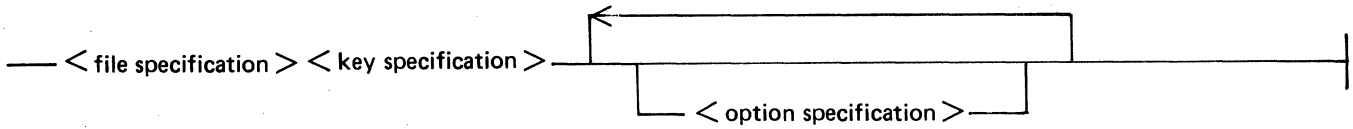
Listing

The SORT program prints a listing of the user-supplied specifications on the line printer. If there are any errors in the specifications, the errors are shown on the listing. The listing also contains a summary of the specifications used by the SORT program after analyzing the user-supplied specifications. The listing can be inhibited by specifying the NOPRINT option.

SORT Specifications

The SORT program specifications consist of a file specification, a key specification and the option specifications. The specifications are entered in free-form format.

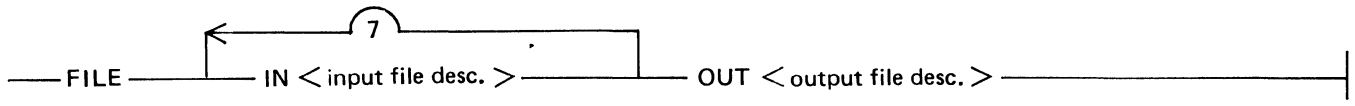
Syntax:



FILE Specification

The FILE specification defines the input and output files of the sort or merge operation. The FILE specification is comprised of the reserved word FILE followed by one to eight input file descriptions and one output file description. Each input file description is preceded by the reserved word IN and the output file description is preceded by the reserved word OUT. If more than one input file description is specified and the DISKSORT option is not specified, then a merge is assumed and the SORT/MERGE intrinsic is invoked.

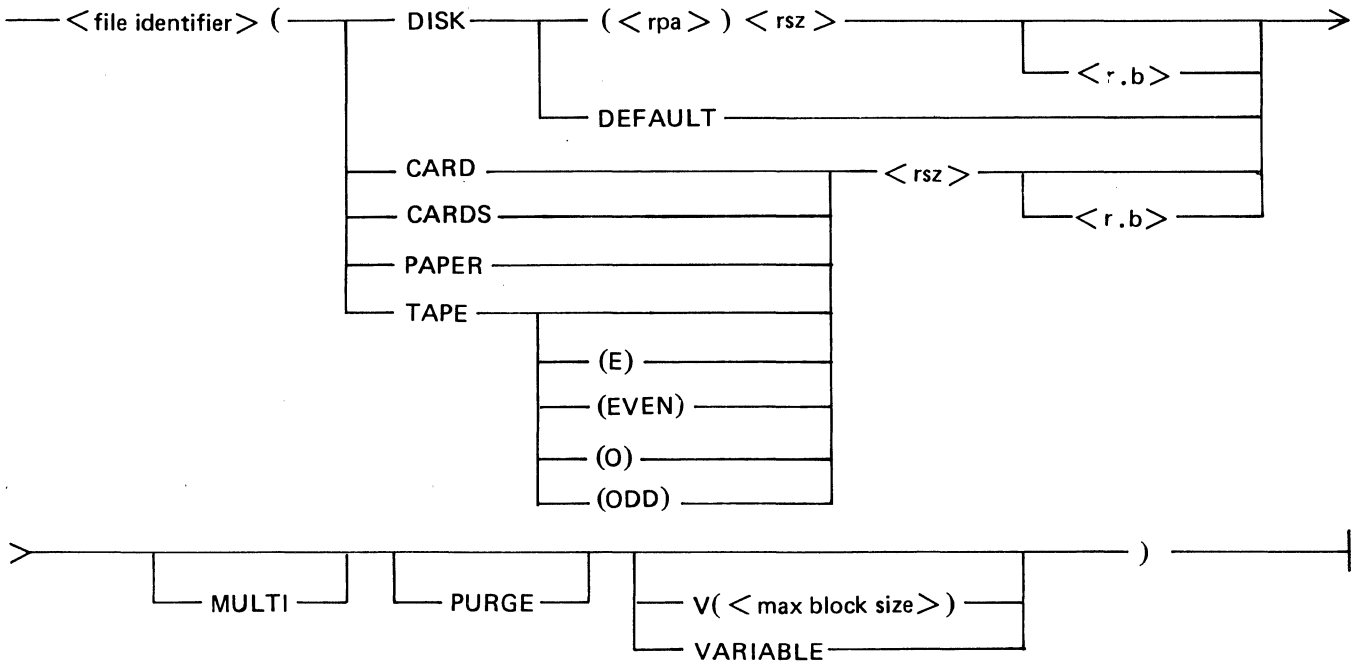
Syntax:



Input File Description

The input file description of the FILE specification describes each input file to be merged or sorted.

Syntax:



G50253

Semantics:

< file identifier >
The < file identifier > specifies the name of the input file.

DISK
DISK specifies that the input file is a disk file.

CARD
CARD specifies that the input file is a card reader file.

CARDS
CARDS is the same as specifying CARD.

PAPER
PAPER specifies that the input file is a paper tape reader file.

TAPE

TAPE specifies that the input file is a tape file.

Seven-track tape is implicitly specified by the inclusion of a parity specifier, enclosed in parentheses, following the tape device-id. ODD or O specifies odd parity, binary mode (no translation). EVEN or E specifies even parity with BCL translation.

DEFAULT

If the input file is on disk, the user may specify DEFAULT which causes the SORT program to obtain the input file specifications (records-per-area, number of areas, record size, and records-per-block) from the disk file header.

<rpa>

When the input file is located on disk and the DEFAULT option is not used, then the number of records-per-area (rpa) must be specified and enclosed in parentheses.

<areas>

Areas is optional and specifies the maximum number of areas on disk that can be allocated for a file.

<rsz>

The record size (rsz) is the actual size in bytes (characters) of the records to be sorted and is a required entry except when the DEFAULT option is specified.

<r.b>

The records-per-block (r.b or blocking factor) is optional and specifies the number of logical records in a block. When the records-per-block is omitted, a default value of 1 applies.

MULTI

The MULTI option specifies that the file is a multipack disk file.

OPTIONAL

OPTIONAL allows the user to specify any input file as optional. If OPTIONAL is specified and the file is not present, then the user is advised by a normal no file message from the MCP. The missing file can be made present or an OF system command can be entered to cause the file to be bypassed. The DEFAULT option cannot be used with OPTIONAL.

PURGE

This option affects input disk and tape files only. If the input file is a disk file, it is removed from the disk directory. If the input file is a magnetic tape file, the tape is purged.

VARIABLE

The VARIABLE or V option allows the user to sort a file of variable length records. When V is specified, the maximum block size must be included. The default value, when VARIABLE is specified, is set equal to record size. Variable length records cannot be sorted when the INPLACE sorting option is specified. The first four characters of each record must contain a 4-digit unsigned alphanumeric integer indicating the length of that record.

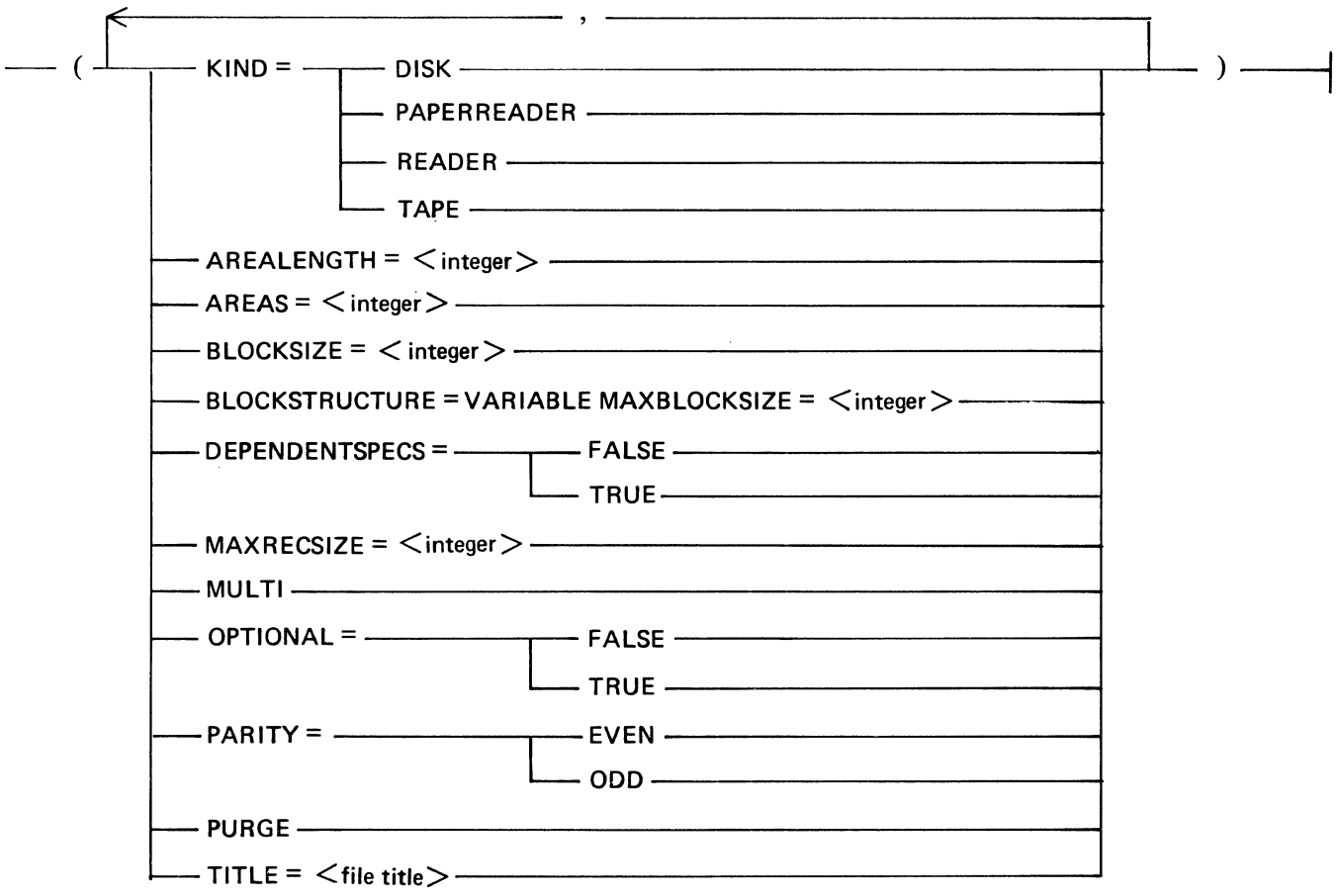
Example:

```
FILE IN DG (TAPE 80 1)
     OUT X/Y/Z (DISK (900 25) 80 9)
```

Input File Description Using File Attributes

The input file description of the FILE specification, which describes one input file to be sorted or merged, can be specified through the use of file attributes. This new file syntax was implemented in the Mark 11.0 software release and is an addition to the previously existing syntax. The new syntax, which makes extensive use of file attributes, is functionally equivalent to the previously existing syntax.

Syntax:



Semantics:

KIND
Specifies the device to be associated with the logical file. Mnemonic values are DISK, PAPER-READER, READER or TAPE.

AREALENGTH
Specifies the number of bytes (characters) in an area of a disk file.

AREAS
Specifies the maximum number of areas that can be allocated for a disk file.

BLOCKSIZE
Specifies the value of the length of a block in bytes (characters).

BLOCKSTRUCTURE
Specifies the format of records of a file. Mnemonic values are FIXED or VARIABLE.

DEPENDENTSPECS
Specifies whether the attributes BLOCKSTRUCTURE, MAXRECSIZE, and BLOCKSIZE of a logical file are to be changed to agree with corresponding values of an associated permanent file. Mnemonic values are TRUE or FALSE.

MAXRECSIZE
Specifies the maximum size of records in bytes (characters).

MULTI
The MULTI option specifies that the file is a multipack disk file.

OPTIONAL
Specifies whether or not the assignment of a permanent file is optional. Mnemonic values are TRUE or FALSE.

PARITY
Specifies the parity used for TAPE or PAPERTAPE files. Mnemonic values are EVEN or ODD.

PURGE
This option affects input disk and tape files only. If the input file is a disk file, it is removed from the disk directory. If the input file is a magnetic tape file, the tape is purged.

TITLE
Specifies the external <file title> in the form B/C ON A. The B/C portion represents the <first name> and the <second name> of the file. The A portion represents the pack <family name>. Valid titles are:

B
B/C
B ON A
B/C ON A

Pragmatics:

1. The TITLE attribute is optional for all files, except DISK files.
2. The KIND attribute is required.
3. The PARITY attribute is for TAPE files only, but is optional.
4. For DISK files, either the DEPENDENTSPECS attribute may be TRUE or the AREALENGTH attribute can be set. DEPENDENTSPECS is equivalent to DEFAULT in the old file syntax.
5. AREALENGTH is equivalent to records per area times record size in the old file syntax.
6. The AREAS attribute is for DISK files only, but is optional. AREAS is not allowed if DEPENDENTSPECS = TRUE.
7. The MAXRECSIZE attribute is required, except when KIND = DISK and DEPENDENTSPECS = TRUE. MAXRECSIZE is equivalent to record size in the old file syntax.
8. The BLOCKSIZE attribute is optional but is not allowed if DEPENDENTSPECS = TRUE. BLOCKSIZE is equivalent to records per block times record size in the old file syntax.
9. The PURGE attribute is optional.
10. The MULTI attribute is for DISK files only and is optional.
11. BLOCKSTRUCTURE is optional.
12. OPTIONAL is not allowed if DEPENDENTSPECS = TRUE.

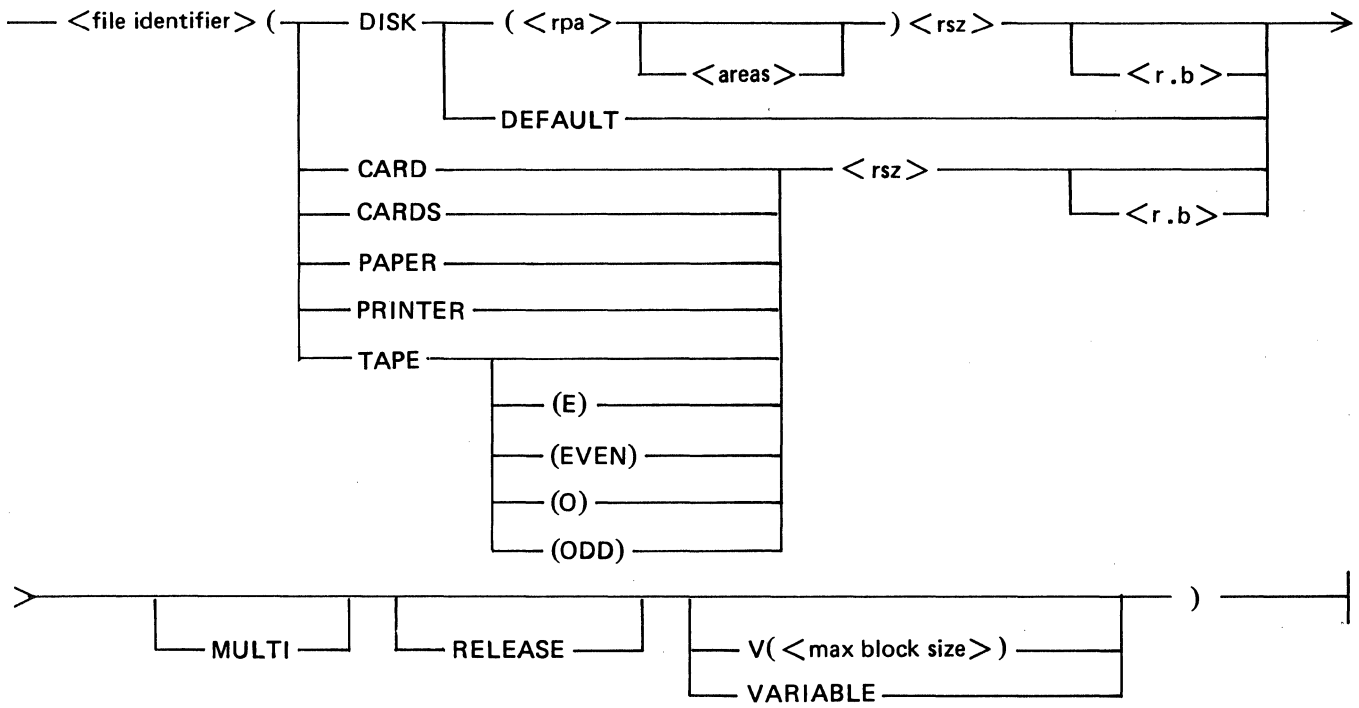
Example:

```
FILE IN (TITLE = ONE,
         KIND = DISK,
         DEPENDENTSPECS = TRUE)
      IN (TITLE = TWO,
         KIND = DISK,
         DEPENDENTSPECS = TRUE)
      OUT (TITLE = THREE,
         KIND = DISK,
         AREALENGTH = 1440,
         AREAS = 1,
         MAXRECSIZE = 80,
         BLOCKSIZE = 160)
KEY (1 5 A UA)
```

Output File Description

The output file description of the FILE specification describes the output file of the sort or merge operation.

Syntax:



G50254

Semantics:

The elements of the output file description have the same relative function to the output file as the elements of the input file description have to the input file, with the exceptions described below.

DISK
DISK specifies that the output file is a disk file.

CARD
CARD specifies that the output file is a card punch file.

CARDS
CARDS is the same as CARD.

PAPER
PAPER specifies that the output file is a paper tape punch file.

PRINTER
PRINTER specifies that the output file is a line printer file.

TAPE
TAPE specifies that the output file is a tape file.

DEFAULT
The DEFAULT option can be specified for the output file only if DEFAULT is also specified for the input file. This option causes the records-per-area, number of areas, record size, and records-per-block of the output file to be identical to those of the input file. DEFAULT cannot be specified when doing a MERGE.

<areas>
Areas specifies the maximum disk areas allowed in the output file.

RELEASE
RELEASE is valid for tape files only and specifies that the tape is to be closed with release.

VARIABLE
This has the same effect on the output file as specifying VARIABLE in the input file description. If VARIABLE is specified in the output file description, it must also be specified in the input file description.

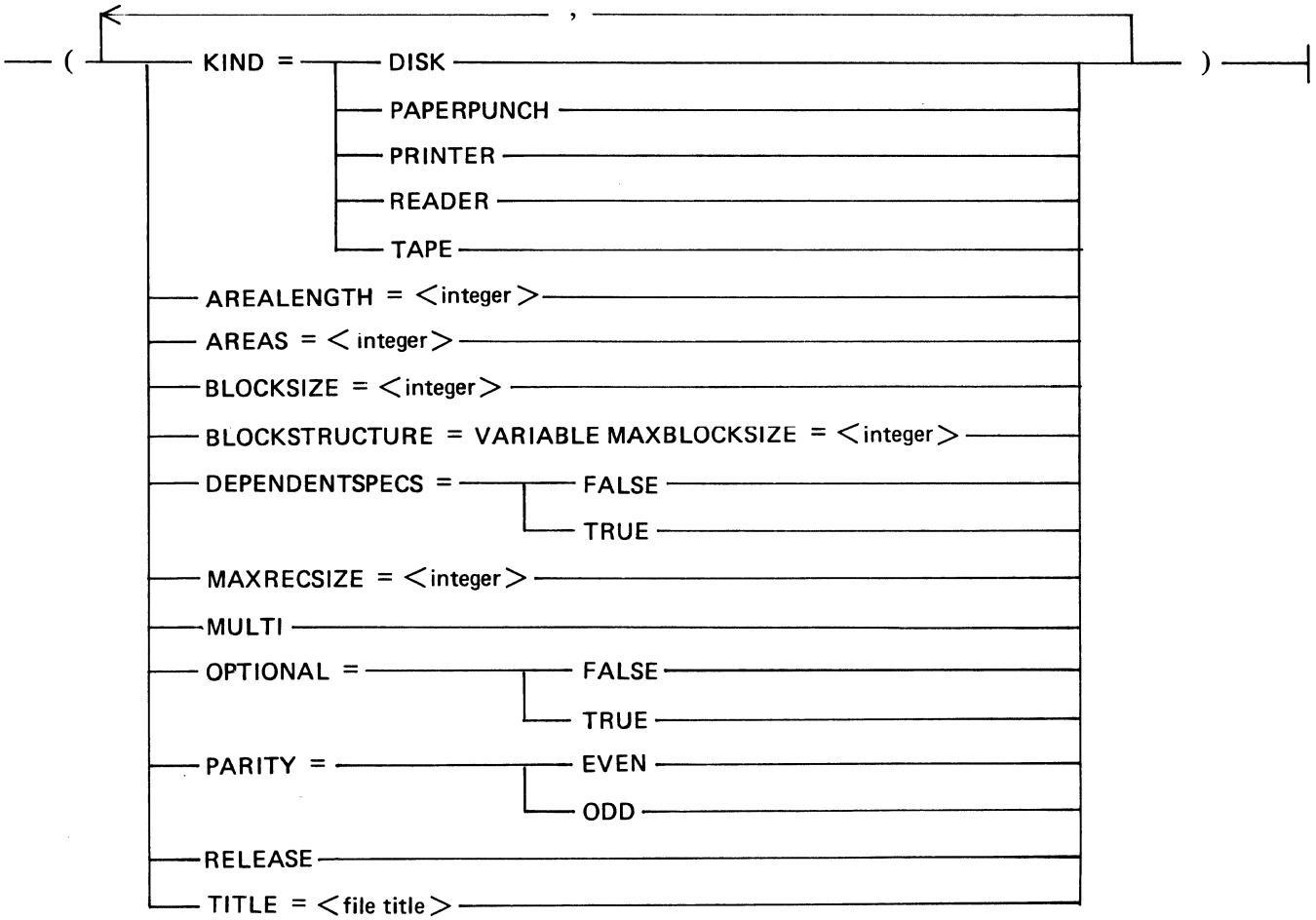
Example:

```
FILE IN DG (TAPE 80 1)
     OUT X/Y/Z (DISK (900 25) 80 9)
```

Output File Description Using File Attributes

The output file description of the FILE specification, which describes the output file of a sort or merge operation, can be specified through the use of file attributes. This new file syntax was implemented in the Mark 11.0 software release and is an addition to the previously existing syntax. The new syntax, which makes extensive use of file attributes, is functionally equivalent to the previously existing syntax.

Syntax:



Semantics:

KIND
Specifies the device to be associated with the logical file. Mnemonic values are DISK, PAPER-PUNCH, PRINTER, READER or TAPE.

AREALENGTH
Specifies the number of bytes (characters) in an area of a disk file.

AREAS
Specifies the maximum number of areas that can be allocated for a disk file.

BLOCKSIZE
Specifies the value of the length of a block in bytes (characters).

BLOCKSTRUCTURE
Specifies the format of records of a file. Mnemonic values are FIXED or VARIABLE.

DEPENDENTSPECS
Specifies whether the attributes BLOCKSTRUCTURE, MAXRECSIZE, and BLOCKSIZE of a logical file are to be changed to agree with corresponding values of an associated permanent file. Mnemonic values are TRUE or FALSE.

MAXRECSIZE
Specifies the maximum size of records in bytes (characters).

MULTI
The MULTI option specifies that the file is a multipack disk file.

OPTIONAL
Specifies whether or not the assignment of a permanent file is optional. Mnemonic values are TRUE or FALSE.

PARITY
Specifies the parity used for TAPE or PAPERTAPE files. Mnemonic values are EVEN or ODD.

RELEASE
RELEASE is valid for tape files only and specifies that the tape is to be closed with release.

TITLE
Specifies the external <file title> in the form B/C ON A. The B/C portion represents the <first name> and the <second name> of the file. The A portion represents the pack <family name>. Valid titles are:

B
B/C
B ON A
B/C ON A

Pragmatics:

1. The TITLE attribute is optional for all files, except DISK files.
2. The KIND attribute is required.
3. The PARITY attribute is for TAPE files only, but is optional.
4. For DISK files, either the DEPENDENTSPECS attribute may be TRUE or the AREALENGTH attribute can be set. DEPENDENTSPECS is equivalent to DEFAULT in the old file syntax.
5. AREALENGTH is equivalent to records per area times record size in the old file syntax.
6. The AREAS attribute is for DISK files only, but is optional. AREAS is not allowed if DEPENDENTSPECS = TRUE.
7. The MAXRECSIZE attribute is required, except when KIND = DISK and DEPENDENTSPECS = TRUE. MAXRECSIZE is equivalent to record size in the old file syntax.
8. The BLOCKSIZE attribute is optional but is not allowed if DEPENDENTSPECS = TRUE. BLOCKSIZE is equivalent to records per block times record size in the old file syntax.
9. The RELEASE attribute is optional. It is not allowed for DISK files.

10. The MULTI attribute is for DISK files only and is optional.
11. BLOCKSTRUCTURE is optional.
12. OPTIONAL is not allowed if DEPENDENTSPECS = TRUE.

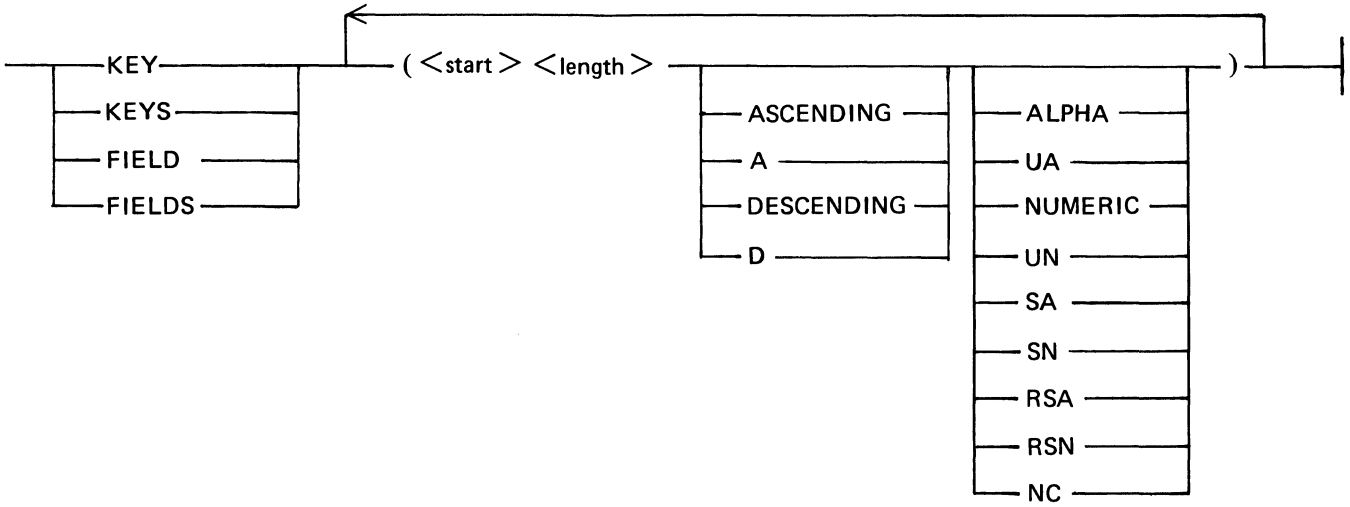
Example:

```
FILE IN (TITLE=ONE,
         KIND=DISK,
         DEPENDENTSPECS=TRUE)
      IN (TITLE=TWO,
         KIND=DISK,
         DEPENDENTSPECS=TRUE)
      OUT (TITLE=THREE,
         KIND=DISK,
         AREALENGTH=1440,
         AREAS=1,
         MAXRECSIZE=80,
         BLOCKSIZE=160)
KEY (1 5 A UA)
```

KEY Specification

The KEY specification defines the field(s) within a record by which the record is to be sorted or merged. The KEY specification is required.

Syntax:



G50255

Semantics:

KEY, KEYS, FIELD or FIELDS
KEY, KEYS, FIELD, and FIELDS are synonyms and may be used interchangeably.

Multiple key descriptions are allowed. The first key is the major key, and any additional keys are minor keys of successive, decreasing significance. Each minor key specified is subordinate to its predecessor.

The maximum number of keys permitted is 30 unsigned keys, or any combination of signed or unsigned keys not exceeding 30, where ALPHA, UA, NUMERIC, UN, and NC count as one, SA, SN, and RSN count as two, and RSA counts as four.

<start>
This must be an integer that specifies the position of the most significant byte or digit of the field including the sign, if any, relative to the first byte or digit of the record. The first byte or digit of a record is considered to be relative position one. The position is referred to in the number of units applicable to the data type of that key.

<length>
The length must be an integer that specifies the number of bytes or digits in the key, including the sign where applicable. The length of a key cannot exceed 511 bytes or 1023 digits.

ASCENDING or A
The key is to be ordered in ascending order, with the lowest key value appearing first. ASCENDING is the default if ASCENDING or DESCENDING is not specified.

DESCENDING or D
Specifying DESCENDING or D results in the keys being ordered in descending order, with the highest key value appearing first.

ALPHA or UA
ALPHA or UA (unsigned alphanumeric) indicates that the data is alphanumeric, and the key location of the field is counted in 8-bit units from the beginning of the record. Alphanumeric is the default if no data type is specified.

NUMERIC or UN
NUMERIC or UN (unsigned numeric) indicates that the data is 4-bit numeric, and the relative position of the field is counted in 4-bit units.

SA
SA (signed alphanumeric) indicates that the data is 8-bit alphanumeric and that some, or all, of the keys can contain a minus sign. The minus sign is represented by a hexadecimal D in the most significant four bits of the first byte of the key.

SN
SN (signed numeric) indicates that the data is 4-bit numeric and that some or all of the keys can contain a minus sign. The minus sign is represented by a hexadecimal D in the first digit of the key.

RSA
RSA (right-signed alphanumeric) means that the data is 8-bit signed alphanumeric, with the minus sign represented by a hexadecimal D in the first four bits of the rightmost byte of the key.

RSN
RSN (right-signed numeric) means that the data is signed 4-bit numeric, and the minus sign is represented by a hexadecimal D in the rightmost digit of the key.

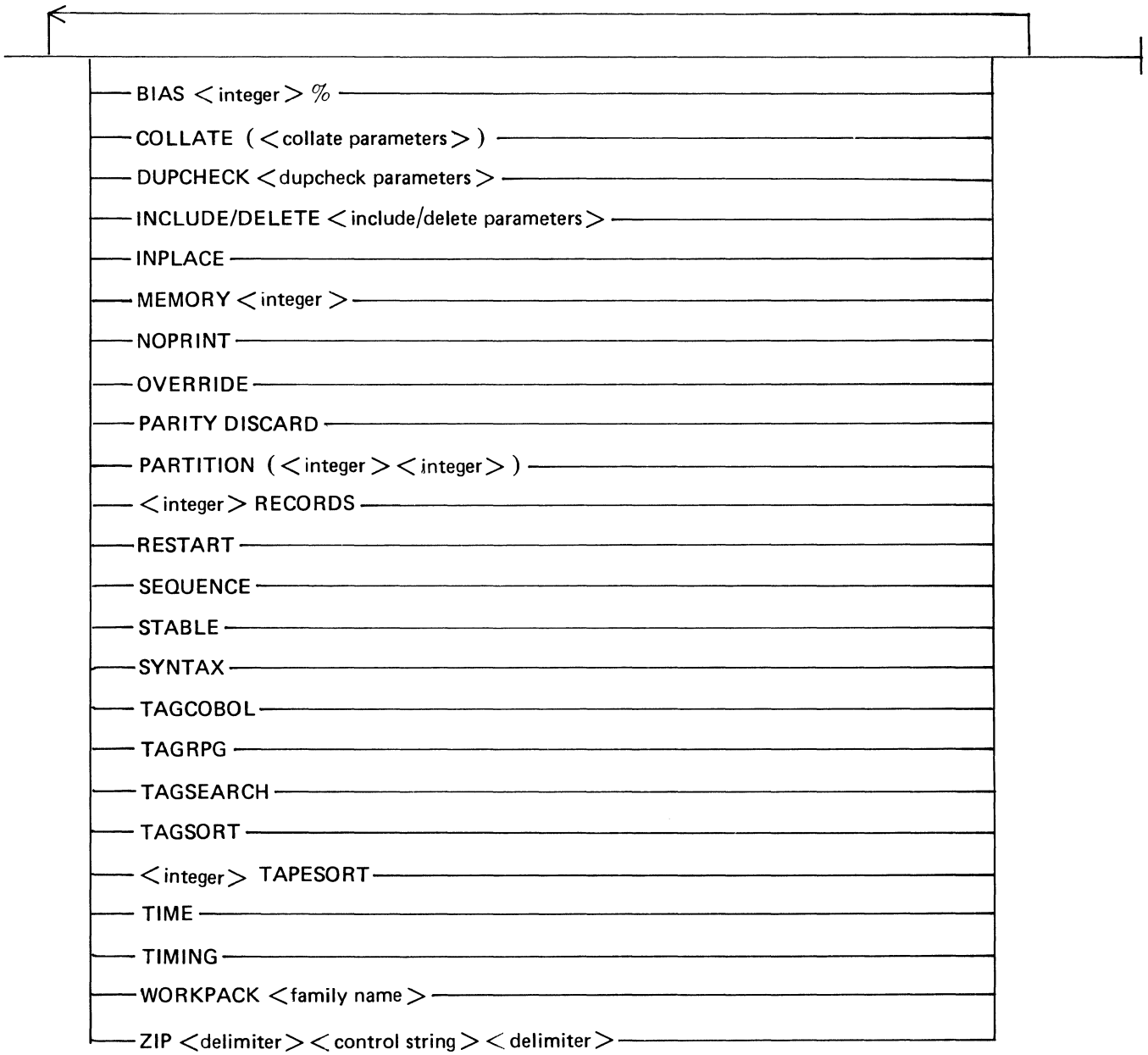
NC
NC (no collate) specifies that the collate file is not to be used for the key (refer to the COLLATE option). The key is assumed to be unsigned alphanumeric.

Option Specifications

The SORT program option specifications have the following functions:

1. Specifying information about the input file, such as the number of records in the file or the bias of the file.
2. Controlling the configuration or operation of the sort.
3. Causing optional operations to be performed before, during, or after the sort execution.

Syntax:



Each of the SORT program option specifications is described in detail in the following paragraphs.

BIAS Option

The BIAS specification is used to indicate to the SORT program the degree of ordering of the records of the input file in relation to the desired output. The estimate is used to optimize sort execution.

The word BIAS must be followed by a number within the range of 0 to 99, where 50 indicates that the records of the input file are in random order and is the default if BIAS is not specified. The number 0 indicates that the records of the file are in reverse order from the sequence desired. The number 97 indicates that the file order is close to the desired sequence.

Example:

```
BIAS 60%
```

The percent sign (%) is optional.

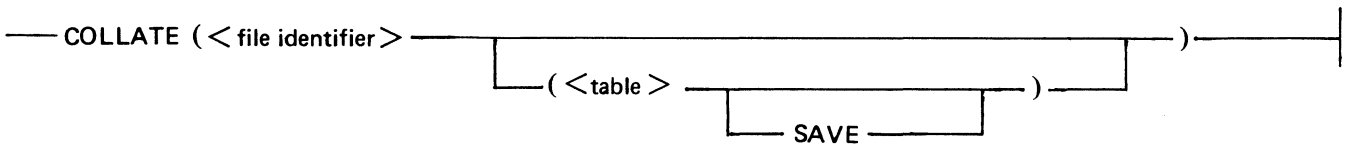
The bias of a file can be determined by specifying the SEQUENCE specification (see the SEQUENCE option).

The BIAS option is only applicable when invoking the SORT/VSORT or SORT/TAPESORT intrinsics.

COLLATE Option

The COLLATE option invokes an optional, virtual collating-sequence capability according to a user-specified collate file. The collate file can be used to specify a new or unique collating sequence, or to retain the standard collating sequence with the exception of certain characters being interchanged or made equal in rank. This option permits the alteration of the sequence in which records are sorted or merged by any of the sort intrinsics. Alteration of the collating sequence may be desired for foreign alphabets or when converting from one processing system to another.

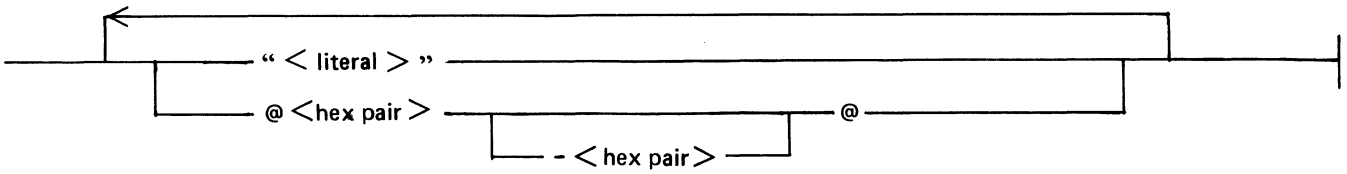
Syntax:



G50257

The <file identifier> is the name of the collate file.

The syntax of the <table> is as follows:



G50258

The <table> that specifies the collate table consists of alphanumeric literals enclosed in quotation marks and hexadecimal strings. Hexadecimal strings can be specified as a range, wherein the hyphen is used as the range specifier (for example, @00-FF@). The first literal or hexadecimal pair collates as @00@, the second one collates as @01@, and so forth.

The generated collate table file is saved on disk, using the name specified by the <file identifier>, if the SAVE option is included; otherwise, the file is purged after use by the SORT program.

If the <table> is not specified, it is assumed to exist on disk with the name specified by the <file identifier>.

Examples:

```
COLLATE(COLLATE/FILE)

COLLATE(TEST/SORTCOLL(@00-C0@"ABCDEFGHI"@CA-D0@
      "JKLMNOPQR"@DA@E1@"STUVWXYZ"@EA-EF@@F0-FF@SAVE))
```

In the first example, the collate file COLLATE/FILE is assumed to exist on disk. In the second example, the specified table is saved on disk as TEST/SORTCOLL. For clarity, the collating sequence in the second example is the default collating sequence to be used if the collate option is not specified in the SORT specifications.

If the specified table contains less than 256 entries, the unspecified entries collate as themselves. Collate table entries always start at @00@.

The collate option affects only those sort keys that are declared unsigned alpha (UA). All other sort keys are sorted in the hardware collating sequence of @00@ through @FF@, which is also the default collating sequence when no collate file is specified.

When the COLLATE option is specified, the MCP sort interface verifies that the collate file is on disk before processing begins. If the collate file is not present, the user is directed to load the file by an operator display terminal(ODT) message. Before opening the collate file, the MCP verifies the disk file header information to ensure that the file consists of two 256-byte records in a single-area file. A message is displayed and the SORT program is discontinued if the collate file is not proper.

The sort or merge intrinsic brings the first record of the collate file into memory and as the key(s) are extracted from each record, the keys that are declared unsigned alpha are processed through a translation operation before being passed to the sort or merge intrinsic comparison logic.

During the final merge phase of the sort intrinsics, the second record of the collate file is used to restore records to their original values.

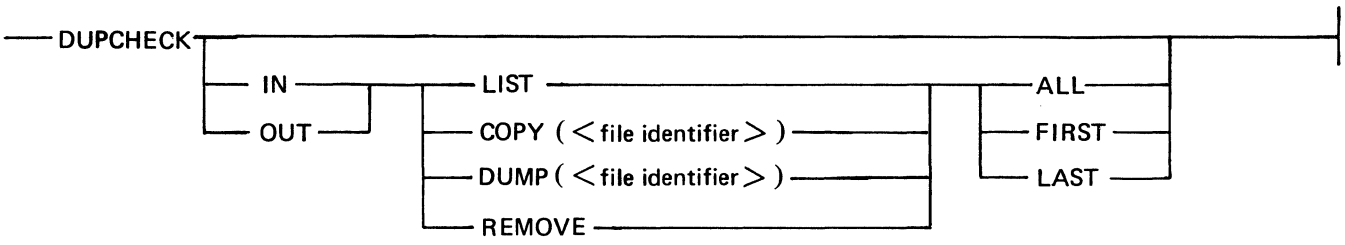
DISKSORT Option

The DISKSORT option causes the SORT/VSORT intrinsic program to be invoked. If the INPLACE, MERGE or TAPESORT options and multiple input files are not specified, then the DISKSORT option is assumed. If multiple input files are specified, the DISKSORT option must be specified or the MERGE option is assumed.

DUPCHECK Option

Duplicate checking is a means of describing an action to be performed upon the occurrence of records that have duplicate keys.

Syntax:



G50261

If DUPCHECK is specified without parameters, duplicate record checking is performed by the sort intrinsic during the output phase. If any duplicate records are detected, a disk file is built consisting of pointers to the records that are duplicates. The sort intrinsic also displays a message on the operator display terminal (ODT) and gives the name of the disk file that contains the pointers. This file, labeled SD.<integer>, consists of 8-byte records. Each record contains two 8-digit (4-byte) numbers. The first number is the relative position of the duplicate record in the output file, and the second number is the relative position of the record that it duplicates. Note that the first record in the output file is considered record number one.

Specifying IN or OUT indicates at what point the duplicate checking is to be performed. If IN is specified, the duplicate checking is performed by the SORT program and no sort intrinsic is called, as the file is assumed to be ordered upon the specified key(s). Except in the case of a copy or a list, where the input file is not altered, a new output file is produced. Specifying OUT causes the output file to be checked in the SORT program after it has been sorted by the sort intrinsic.

Specifying LIST causes the duplicate records to be listed on the line printer. COPY creates a file with the name specified by the <file identifier> and that file contains a copy of the duplicate records. Neither the LIST nor the COPY options cause records to be removed from the file. REMOVE causes the duplicate records to be removed and discarded. DUMP builds a file of the removed records under the name specified by the <file identifier>. The files created by the COPY and DUMP options have the same attributes as the input file.

If ALL is specified, the set of duplicate records consists of all the subsets of records that are duplicates of each other. If FIRST is specified, the set of duplicate records consists of all records in all subsets except the first record in each subset. Likewise, LAST causes the set of duplicate records to consist of all records in all subsets except the last record in each subset.

If the COLLATE and DUPCHECK options are both specified, the duplicate checking is performed while the file is in the uncollated state.

DUPCHECK with either IN or OUT cannot be specified when a sort is being compiled.

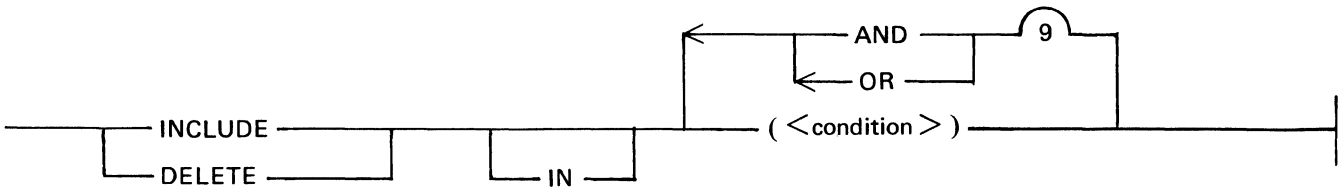
Example:

```
DUPCHECK OUT DUMP (A/B/C) FIRST
```

INCLUDE/DELETE Option

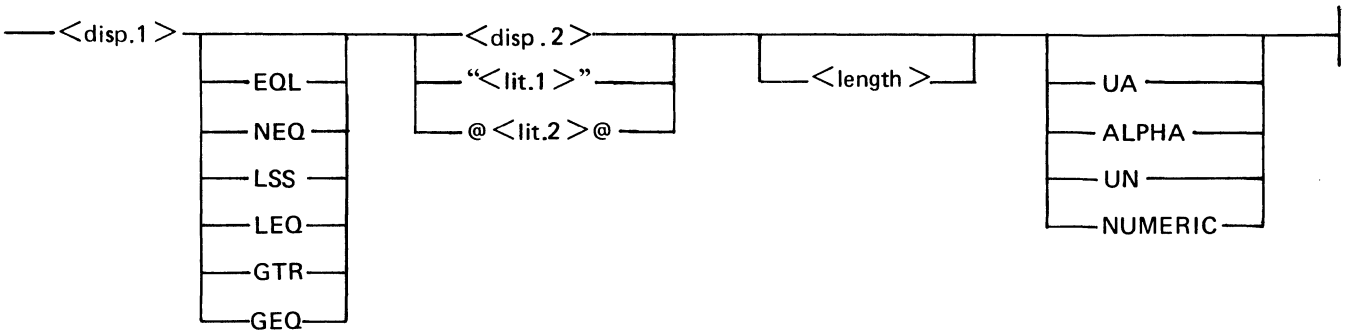
The INCLUDE/DELETE option allows the inclusion or exclusion of certain records in a file, according to the relational evaluations of portions of the record with other portions of the same record or with specified literals. The INCLUDE/DELETE function is available in the SORT/VSORT and SORT/TAPESORT intrinsics; this function is also available as a stand-alone function of the SORT program.

Syntax:



G50259

The syntax of the <condition> is as follows:



G50260

The reserved words INCLUDE and DELETE can appear only once within a set of sort specifications, but each word can be followed by a plurality of keys. The total number of keys must not exceed ten. Multiple keys within an INCLUDE or DELETE are joined by the reserved words AND or OR. The resulting expressions are evaluated, element by element, left to right. If an OR is encountered and all previous expression elements are satisfied, then the evaluation is terminated as true. If an AND is encountered and all previous expression elements are not satisfied, then the evaluation is terminated as false. In all other circumstances, the evaluation is continued. Another way of looking at the evaluation follows:

(A r (B r (C r (D))))

Where A, B, C, and D are conditions, and r is a relation (AND or OR). The parentheses are used only to illustrate precedence.

When both INCLUDE and DELETE specifications are entered for a sort, a record may be specified to be both deleted and included. In this case, the last specification is followed.

Specifying IN causes the INCLUDE/DELETE option to be performed within the SORT program, and produces a file written according to the output specifications. No sort intrinsic is invoked. If COLLATE is specified, the INCLUDE/DELETE function is performed while the file is in the uncollated state.

If IN is not specified, the INCLUDE/DELETE function is performed within the appropriate sort intrinsic and, if COLLATE is also specified, after the file is collated. Note that the relationship of order between the COLLATE option and the INCLUDE/DELETE option is dependent upon whether it is performed in the SORT program or in one of the intrinsics. IN cannot be specified when compiling a sort.

<Disp.1> specifies the offset of the key (first operand) from the beginning of the record. The offset is in bytes or digits, depending upon the ALPHA or NUMERIC entry.

The entry following the key displacement is the relational operator to be applied to the comparison of the specified operands. Valid entries are EQL (equal), NEQ (not equal), GTR (greater), GEQ (greater or equal), LSS (less), and LEQ (less than or equal). The default, if no entry is specified, is equal.

The second operand (disp.2) may be specified as a displacement into the record or as an alphanumeric or hexadecimal literal. As a displacement, the second operand is subject to the rules outlined for the first operand. Hexadecimal literals are delimited by at signs (@) and may contain one to six hexadecimal digits. The maximum key length is 3. Therefore, when using a numeric field the maximum size to be used is three hexadecimal digits. Character literals are delimited by quotation marks and may contain up to three characters. All characters except the quotation mark or question mark are valid. Literals are padded or truncated as necessary, according to the length attribute.

The length field specifies the length of the operands to be compared. This length is 8-bit or 4-bit depending on the ALPHA or NUMERIC entry. The default, if no length is specified, is 1. The maximum allowable length is 3. If the hexadecimal literal <lit.2> entered is greater than the length specified, a syntax error is generated.

ALPHA or NUMERIC specify whether the operands are character or digit oriented. These entries can be abbreviated UA and UN, respectively. Default is ALPHA if nothing is specified.

Example:

```
INCLUDE (1 EQL "DG" 2 UA) AND (5 EQL "SG" 2 UA)
```

INPLACE Option

The INPLACE option is advantageous when only a minimum amount of disk space is available for sorting. The INPLACE option invokes the SORT/QSORT intrinsic program. The INPLACE sort requires work file space equal to the input file space, unless the input and output file identifiers are the same and both are disk files. In that case, no work file space is required, but the input file is overlaid by the output file during the sorting process. The maximum record size that can be sorted is 2730 bytes. In addition, the following expression

```
2 * (record size * records-per-block) + (record size)
```

must be less than or equal to 8192 bytes.

MEMORY Option

The MEMORY or ME option is used to allocate more memory to the sort than the 8000 bytes assigned by default. The number of bytes must follow the word MEMORY or ME and be within the range of 8000 to 125000. The maximum is 18500 when the INPLACE option is specified.

Increasing the memory available to the sort is the most significant means of increasing sort efficiency, up to an optimum sort memory size. Increasing the memory size beyond the optimum size does not cause the sort to run faster and may actually cause it to run slower.The optimum size varies depending on file size (record size, blocking factor, and number of records) and other jobs in the mix at the time of the sort. To determine the optimum size for any given file, the user should experiment with different memory sizes.

Examples:

```
MEMORY 12000
ME 24000
```

MERGE Option

The MERGE option causes the SORT/MERGE intrinsic program to be invoked. If multiple input files are specified and the DISKSORT or TAPESORT option is not specified, then the MERGE option is assumed.

NOPRINT Option

The NOPRINT option is used to inhibit the printing of the SORT specifications on the line printer, and if used, must be the first entry in the SORT program specifications.

Example:

```
NOPRINT
```

OVERRIDE option

When the input file is on disk and the file attributes specified do not match the actual file attributes, then the sort program is DS-ed automatically. If OVERRIDE is specified, the sort intrinsic program is not DS-ed, the actual attributes of the file are ignored, and the specified attributes are used.

Example:

```
OVERRIDE
```

PARITY DISCARD Option

The PARITY DISCARD option causes input file records to be discarded if an irrecoverable parity error is detected during the reading of the input file. If PARITY DISCARD is specified, the block of data in which the error occurs is discarded and notification for each record lost, along with its relative record position in the input file and is displayed on the operator display terminal (ODT). If PARITY DISCARD is not specified and a parity error occurs, then the program is discontinued.

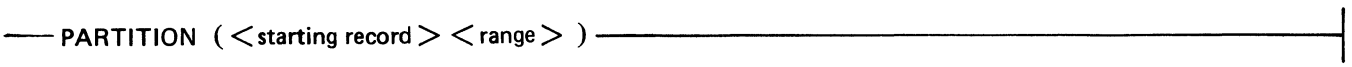
Example:

```
PARITY DISCARD
```

PARTITION Option

The PARTITION option, available when invoking the SORT/VSORT or SORT/TAPESORT intrinsics, allows any specified portion of the input file to be sorted. Partitioning occurs before any other file manipulations.

Syntax:



G50262

The number of the first record in the file to be sorted is specified by <starting record>. The first record in the file is counted as number 1. The <range> indicates the number of records to be sorted.

Only one PARTITION option may be included in a sort specification deck.

Example:

```
PARTITION (500 700)
```

RECORDS Option

The RECORDS option is used to optimize sort operation by supplying an estimate of the total number of records in the input file or files. The estimated number must be within the range of 0 to 8388608 and must precede the word RECORDS. If this option is omitted, a default value of 20,000 is used. If the input file is on disk, the number of records is obtained from the disk file header, regardless if this option is specified.

Example:

```
12500 RECORDS
```

RESTART Option

A sort program which is abnormally terminated may be restarted at a point following the latest successful sort pass:

1. RESTART must be specified in the set of sort specifications if the ensuing sort is to be considered restartable.
2. After abnormal termination, the sort may be restarted by inserting a RESTART <job number> statement into the sort specification deck and re-executing the sort. The <job-number> specified is the job number of the program that called the sort intrinsic.
3. If the sort is a compiled sort, the sort codefile may be executed with the program switches set to the <job number>, as follows:

   SW = <job number>

The RESTART feature is not available with the INPLACE sort or when merging.

Examples:

```
RESTART
RESTART 1259
EX SORTPROG SW = 1259
```

SEQUENCE Option

The SEQUENCE option checks the sequence of the input file according to the specified key or keys. The number of records in the file, the number of sequence errors, and the bias are displayed on the operator display terminal (ODT).

The bias is calculated as follows:

BIAS = 99-(number of sequence errors X 100)/number of records)

The calculated bias is rounded to the nearest integer.

Sort intrinsics are not invoked when SEQUENCE is specified. SEQUENCE must not be specified when compiling a sort.

Example:

```
SEQUENCE
```

STABLE Option

The STABLE option which is allowed only with the SORT/VSORT or SORT/TAPESORT intrinsics causes records that are equal (records which have duplicate key fields) to be placed into the sorted output file in the same order in which they occurred in the input file. If STABLE is not specified, the order of such duplicate records in the output file cannot be predicted.

An extra (implicit) key is appended to the record by the sort intrinsic during the input phase to be used as the "tie-breaker" for records that have equal keys. This implicit key (the input file relative record number) is removed during the output phase.

SYNTAX Option

The SYNTAX option is used when the SORT specifications are to be checked for errors only. The sort intrinsic is not executed.

Example:

```
SYNTAX
```

TAGCOBOL Option

Specifying TAGCOBOL causes a tagsort to be performed. The resultant output file is created in a format compatible with the index file used by the COBOL '68 Index-Sequential Access Method. Each record in the output file contains keys from the input record, concatenated together and followed by the 8-digit (4-byte) numeric record pointer. Thus, the output file record size is declared as the sum of the length of all the key fields plus four bytes.

Example:

```
TAGCOBOL
```

TAGRPG Option

Specifying TAGRPG causes a tagsort to be performed, with the resultant output file created in a format compatible with the tag file used by the RPG indexed ($TAG) access method. Each record in the output file contains the keys from the input record concatenated together, followed by the 6-digit (3-byte) numeric record pointer. Thus, the output file record size must be declared to be the sum of the lengths of all the key fields plus three bytes.

Example:

```
TAGRPG
```

TAGSEARCH Option

Specifying TAGSEARCH causes a tagsort to be performed. The resultant sorted file of indices is used to build a sorted output file. Depending on the characteristics of the file and the number of records to be sorted, TAGSEARCH can attain a greater overall sorting speed than other available sorting methods. When sorting variable length records, do not specify TAGSEARCH.

Example:

```
TAGSEARCH
```

TAGSORT Option

The TAGSORT option provides a means of sorting a file and creating an output file containing indices that point to the relative locations of records within the original file. The input file remains intact.

Input files can originate from any allowable hardware device type, but must reside on disk when using the file of indices to access the actual records.

The output file, referred to as the tag file, must be defined as four characters per record because each record consists of an 8-digit numeric number, used as an index that points to the corresponding record in the input file.

TAGSORT is allowed only when invoking the SORT/VSORT or SORT/TAPESORT intrinsics.

Access to the input file is provided using the tag file as follows:

1. When sorting in COBOL, the specified access method is RANDOM and the tag-file record is used as the ACTUAL KEY.
2. When sorting in RPG, the specified access method is INDEXED and the relative record number is used as delivered in the tag file to directly (DIRECT) access the original file.

Example:

```
TAGSORT
```

TAPESORT Option

The SORT/TAPESORT intrinsic program is invoked if <integer> TAPESORT is specified. The number of work files on magnetic tape can range from three to eight and is specified by <integer>.

The work files are opened on any available tape units. If tape output is requested, the first tape unit accessed as a work file is of the same tape hardware type as specified for the output file. The sort algorithm causes this tape unit to be exhausted of data records on the merge pass that precedes the final merge pass, thereby making that tape unit available to receive the sorted output file.

All writing of work-file tapes is done in a forward direction, and all reading of work-file tapes is done in reverse direction; therefore, multiple tape reels are not allowed for any individual work file. The user must use work-file tape reels of sufficient size to hold all of the records assigned to it by the SORT/TAPESORT intrinsic program. The number of records assigned to a work file is less the number of records in the original input file, but can approach that number. For multiple reel input files, each tape reel can be sorted separately, and the resultant output files can then be merged using the SORT/MERGE intrinsic.

The following example specifies that five work files are to be used for the SORT/TAPESORT intrinsic.

Example:

```
5 TAPESORT
```

A self-checking feature of TAPESORT aids in guaranteeing the integrity of the data. As each block of data is written on a work tape, it is concatenated with the contents of an incremented counter. Thus, each block contains a number representing its relative position on the tape. Upon reading the work tape, these numbers are checked for incremental sequence.

TIME Option

At the end of the SORT/VSORT intrinsic execution, if TIME is specified, then the number of records sorted, the elapsed time (ET) for the sort, the processor time (PT) used by SORT/VSORT, and the number of sort passes made is displayed on the operator display terminal (ODT).

Example:

```
TIME
```

TIMING Option

The TIMING option, which may be specified when the SORT/VSORT intrinsic is used, provides the following information on the line printer:

1. Distribute, Merge, and Final Merge Vector size (VECTOR) in records.
2. Input, output, and work file attributes.
3. A summary of the sort parameters and sort options specified.

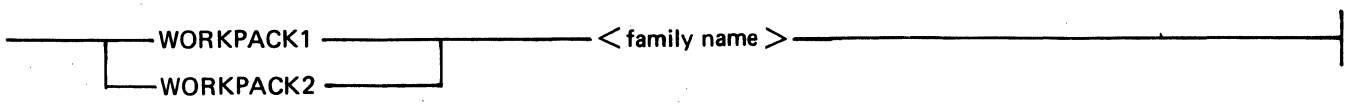
Example:

```
TIMING
```

WORKPACK Option

It is possible to designate specific user disks for use by the two SORT/VSORT intrinsic work files, W1 and W2. Increased throughput can be achieved by careful assignment of the work files.

Syntax:



G50263

Both WORKPACK options may be declared in one set of sort specifications. The WORKPACK options are only valid when invoking the SORT/VSORT intrinsic. The <family name> is the name of the disk or pack.

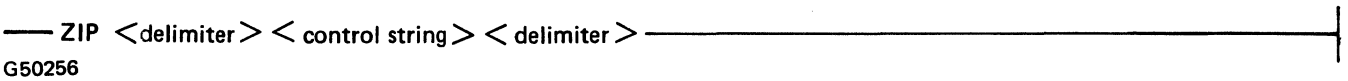
Examples:

```
WORKPACK1 USER1
WORKPACK2 USER2
```

ZIP Option

The ZIP option allows a control string to be passed (zipped) to the MCP before the sort program goes to end of job.

Syntax:



G50256

The delimiters can be any non-blank character. The first non-blank character following the reserved word ZIP is used as the beginning delimiter. The control-string is terminated at the next occurrence of the delimiter character; thus, the delimiter character cannot be used within the control-string itself. If the beginning delimiter is a left parenthesis, left bracket, or left brace, then the ending delimiter must be a right parenthesis, right bracket, or right brace, respectively. For all other characters, the ending delimiter must be the same as the beginning delimiter.

Only one ZIP statement, limited to a maximum length of 255 characters, can be contained in the SORT specifications.

Examples:

```
ZIP (EX A/B AFTER SORT FILE DISK NAME C/D;)
ZIP "USER=SITE EX DMPALL PR 6"
```

SORT Reserved Words and Characters

A list of the SORT program reserved words and characters follows. The reserved words and characters in this list cannot be used in comments, unless the comment is preceded by a colon (:).

BIAS	KEY	SEQUENCE
	KEYS	STABLE
COLLATE		SYNTAX
	ME	
DELETE	MEMORY	TAGCOBOL
DISKSORT	MERGE	TAGRPG
DUPCHECK		TAGSEARCH
	NOPRINT	TAGSORT
FIELD		TAPESORT
FIELDS	OVERRIDE	TIME
FILE		TIMING
	PARITY	
	PARTITION	WAIT
	PASSPARAM	WORKPACK1
INCLUDE		WORKPACK2
INPLACE	RECORDS	
	RESTART	ZIP
		(

SECTION 3
SORT/UTILITY

The SORT/UTILITY program is used to sort and merge files.

Selected records of the input file may be processed. Depending on selected conditions, records of an input file may be included or excluded from sort processing.

The format and contents of the output records may be changed from the input format. Fields in a record may be rearranged or excluded. Selected fields may be summed together to provide user processing during the sort run. Literals may be inserted into chosen positions in the output records.

Different record formats may be specified within the same output file. Selected criteria determine the arrangement and selection of fields in a record; there may be numerous sets of criteria for one output file.

The order of the output file may be controlled by the user. Records may be sorted in ascending or descending order. The normal collating sequence can be altered.

The SORT/UTILITY program has the ability to process right-or left-signed, unpacked or packed (character or numeric) data. Also, zone or digit portions of characters may be specified.

The SORT/UTILITY program performs all of the same functions as the IBM $DSORT program. Specifications may be given in free-form format, as in the SORT program, or in the fixed field format of $DSORT.

The SORT/UTILITY program, after analyzing the sort specifications, copies to a temporary disk file any of the input files that are not on disk. The SORT/UTILITY program then extracts the keys, called the control field, from the input file or files and builds a work file. The work file consists of these keys and a pointer indicating the input file and the record number. Next, the file is sorted using the SORT/VSORT intrinsic program. Finally, the SORT/UTILITY program uses the work file to perform a tagsearch on the input files and to create the output file.

SORT/UTILITY Program Execution

The procedures used to execute the SORT/UTILITY program are explained in the following paragraphs.

Card Reader Input

The SORT/UTILITY program execution card deck consists of an EXECUTE card, a DATA card, the sort specification cards, and an END card.

Example:

```
? EXECUTE SORT/UTILITY
? DATA CARDS
(specification cards)
? END
```

ODT Keyboard Input

If the SORT/UTILITY program is executed with switch 0 or switch 8 set to 1, then the card input file is not used and specifications are entered from the Operator Display Terminal (ODT) as a series of ACCEPT (AX) messages. A blank (null) ACCEPT message terminates the series and begins the sort procedure.

Comments

A colon (:) terminates the scanning of a sort specification record. Any comment may follow a colon. The colon is ignored as a comment indicator if it appears in a ZIP statement, file identifier, or within quotation marks.

Example:

```
:THIS IS A COMMENT
FILE IN A/RANDOM (DISK DEFAULT)
FILE OUT A/ORDERED (DISK DEFAULT) : AND THIS IS A COMMENT
KEY (68 2 A ALPHA)
```

Compiling Sort Programs

The sort program is either executed immediately or may be compiled to library for later execution. If the SORT/UTILITY program is executed by an EXECUTE control instruction, the specifications are processed and the sorting is immediately invoked provided that no syntax errors are detected. However, if the SORT/UTILITY program is executed by a COMPILE TO LIBRARY control instruction, an object program is compiled to the disk library for execution at a later time. For example, the following control string compiles an object sort program named SORTPROG to disk:

```
COMPILE SORTPROG WITH SORT/UTILITY TO LIBRARY
```

At any time thereafter the compiled sort program may be executed.

Example:

```
EXECUTE SORTPROG
```

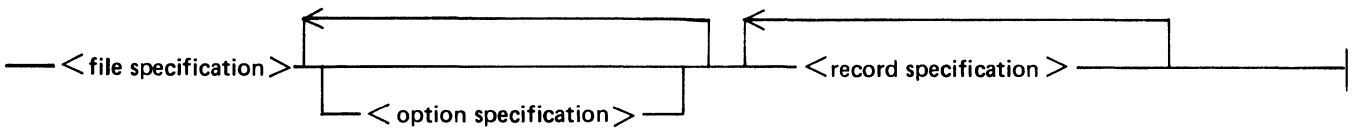
For more information on the EXECUTE or COMPILE control instructions, refer to the B 1000 Systems System Software Operation Guide, Volume 1, form number 1151982.

The SORT/UTILITY program stores the sort specifications in its dynamic memory. If the dynamic memory size is not large enough to hold the specifications, a message is displayed indicating the problem. The user should re-execute the SORT/UTILITY program and increase the dynamic memory size appropriately.

Free-Form Format Specifications

The use of the free-form format specifications is described in this subsection. The free-form format syntax includes some capabilities that the fixed format doesn't have. Character constants are restricted only by record size in the free-form format. The unsigned numeric (UN) data type is specified only in the free-form format. The ZIP, RESTART, TIMING, and STABLE options are specified only in the free-form format syntax. The free-form format specifications consist of a file specification, the option specifications and the record specifications.

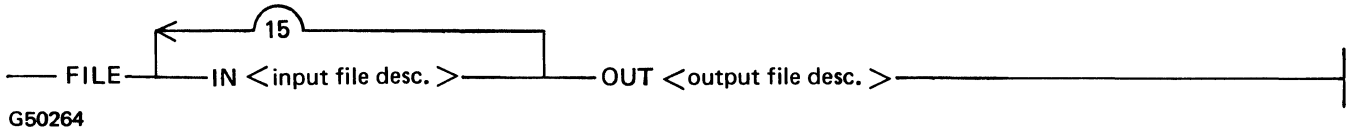
Syntax:



FILE Specification

The FILE specification defines the input and output files of the sort or merge operation. The FILE specification is comprised of the reserved word FILE followed by one to sixteen input file descriptions and one output file description. Each input file description is preceded by the reserved word IN and the output file description is preceded by the reserved word OUT. If more than one input file description is specified, then a merge is assumed and the SORT/MERGE intrinsic is invoked.

Syntax:

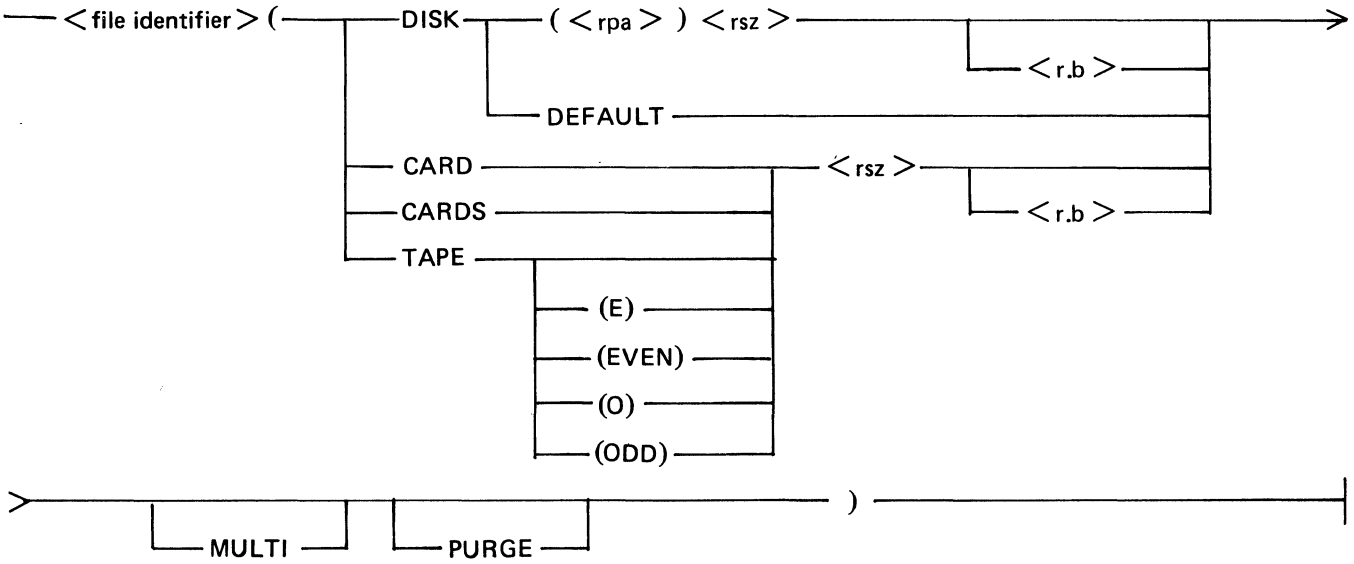


G50264

Input File Description

The input file description of the FILE specification describes each input file to be merged or sorted.

Syntax:



G50265

Semantics:

<file identifier>
The <file identifier> specifies the name of the input file.

DISK
DISK specifies that the input file is a disk file.

CARD
CARD specifies that the input file is a card reader file.

CARDS
CARDS is the same as specifying CARD.

TAPE
TAPE specifies that the input file is a tape file.

Seven-track tape is implicitly specified by the inclusion of a parity specifier, enclosed in parentheses, following the tape device-id. ODD or O specifies odd parity, binary mode (no translation) EVEN or E specifies even parity with BCL translation.

DEFAULT
If the input file is on disk, the user may specify DEFAULT which causes the SORT program to obtain the input file specifications (records-per-area, number of areas, record size, records-per-block) from the disk file header.

<rpa>
When the input file is located on disk and the DEFAULT option is not used, then the number of records-per-area (rpa) must be specified and enclosed in parentheses.

<areas>
Areas is optional and specifies the maximum number of areas on disk that can be allocated for a disk file.

<rsz>
The record size (rsz) is the actual size in bytes (characters) of the records to be sorted and is a required entry except when the DEFAULT option is specified.

<r.b>
The records-per-block (r.b or blocking factor) is optional and specifies the number of logical records in a block. When the records-per-block is omitted, a default value of 1 applies.

MULTI
The MULTI option specifies that the file is a multipack disk file.

PURGE
This option affects input disk and tape files only. If the input file is a disk file, it is removed from the disk directory. If the input file is a magnetic tape file, the tape is purged.

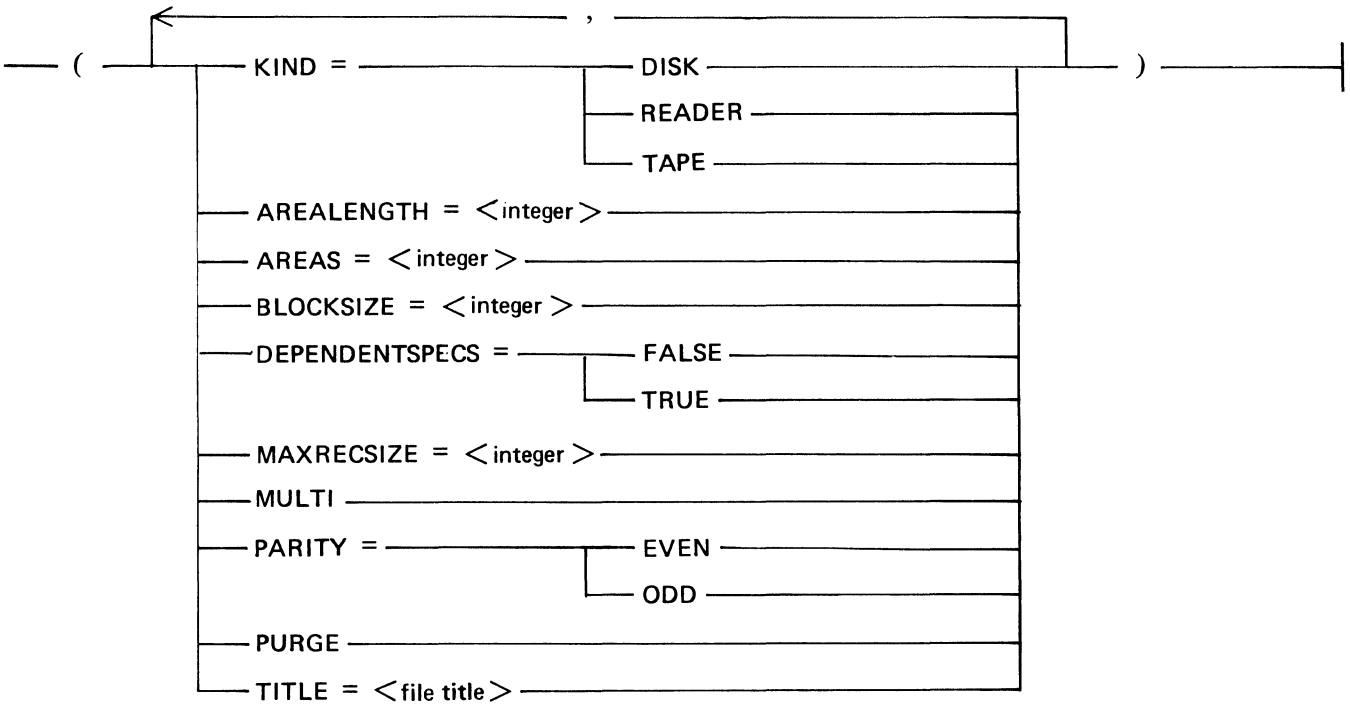
Example:

```
FILE IN DG (TAPE 80 1)
     OUT X/Y/Z (DISK (900 25) 80 9)
```

Input File Description Using File Attributes

The input file description of the FILE specification, which describes one input file to be sorted or merged, can be specified through the use of file attributes. This new file syntax was implemented in the Mark 11.0 software release and is an addition to the previously existing syntax. The new syntax, which makes extensive use of file attributes, is functionally equivalent to the previously existing syntax.

Syntax:



Semantics:

KIND
Specifies the device to be associated with the logical file. Mnemonic values are DISK, READER or TAPE.

AREALENGTH
Specifies the number of bytes (characters) in an area of a disk file.

AREAS
Specifies the maximum number of areas that can be allocated for a disk file.

BLOCKSIZE
Specifies the value of the length of a block in bytes (characters).

DEPENDENTSPECS
Specifies whether the attributes BLOCKSTRUCTURE, MAXRECSIZE, and BLOCKSIZE of a logical file are to be changed to agree with corresponding values of an associated permanent file. Mnemonic values are TRUE or FALSE.

MAXRECSIZE
Specifies the maximum size of records in bytes (characters).

MULTI
The MULTI option specifies that the file is a multipack disk file.

PARITY
Specifies the parity used for TAPE files. Mnemonic values are EVEN or ODD.

PURGE
This option affects input disk and tape files only. If the input file is a disk file, it is removed from the disk directory. If the input file is a magnetic tape file, the tape is purged.

TITLE
Specifies the external <file title> in the form B/C ON A. The B/C portion represents the <first name> and the <second name> of the file. The A portion represents the pack <family name>. Valid titles are:

B
B/C
B ON A
B/C ON A

Pragmatics:

1. The TITLE attribute is optional for all files, except DISK files.
2. The KIND attribute is required.
3. The PARITY attribute is for TAPE files only, but is optional.
4. For DISK files, either the DEPENDENTSPECS attribute may be TRUE or the AREALENGTH attribute can be set. DEPENDENTSPECS is equivalent to DEFAULT in the old file syntax.
5. AREALENGTH is equivalent to records per area times record size in the old file syntax.
6. The AREAS attribute is for DISK files only, but is optional. AREAS is not allowed if DEPENDENTSPECS = TRUE.
7. The MAXRECSIZE attribute is required, except when KIND = DISK and DEPENDENTSPECS = TRUE. MAXRECSIZE is equivalent to record size in the old file syntax.
8. The BLOCKSIZE attribute is optional but is not allowed if DEPENDENTSPECS = TRUE. BLOCKSIZE is equivalent to records per block times record size in the old file syntax.
9. The PURGE attribute is optional.
10. The MULTI attribute is for DISK files only and is optional.

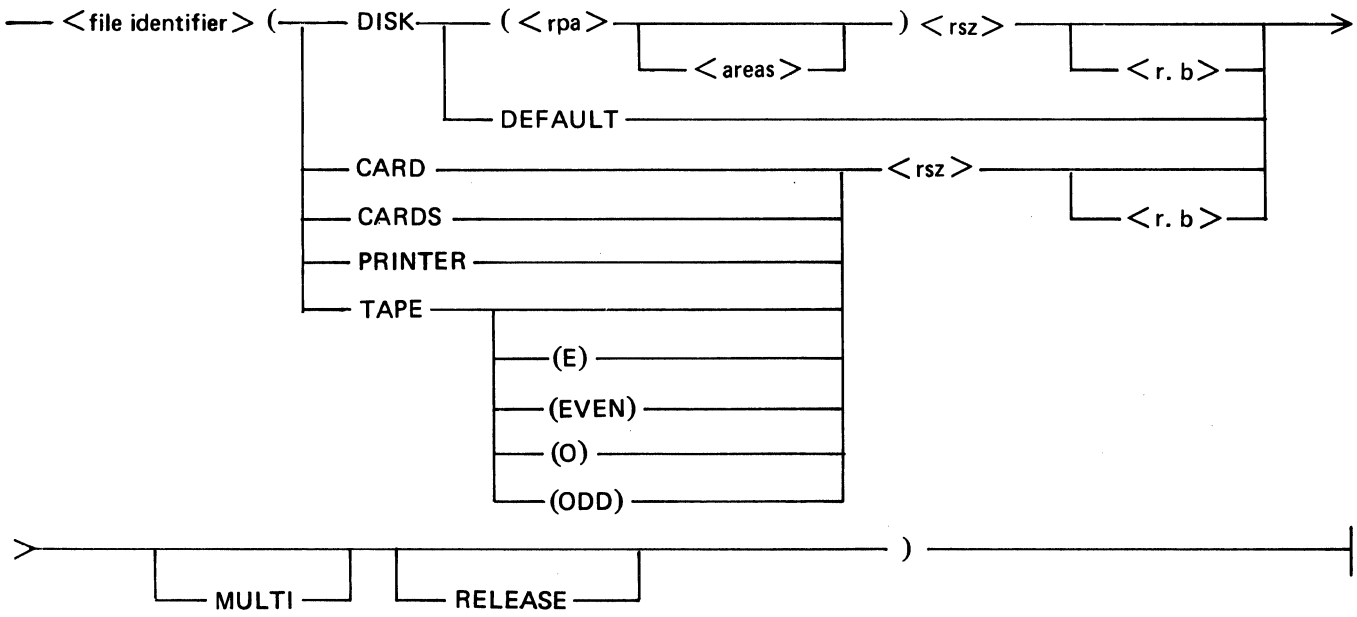
Example:

```
FILE IN (TITLE=ONE,
         KIND=DISK,
         DEPENDENTSPECS=TRUE)
     IN (TITLE=TWO,
         KIND=DISK,
         DEPENDENTSPECS=TRUE)
     OUT (TITLE = THREE,
         KIND=DISK,
         AREALENGTH=1440,
         AREAS=1,
         MAXRECSIZE = 80,
         BLOCKSIZE=160)
KEY (1 5 A UA)
```

Output File Description

The output file description of the FILE specification describes the output file of the sort or merge operation.

Syntax:



Semantics:

The elements of the output file description have the same relative function to the output file as the elements of the input file description have to the input file, with the exceptions described below.

DISK
DISK specifies that the output file is a disk file.

CARD
CARD specifies that the output file is a card punch file.

CARDS
CARDS is the same as CARD.

PRINTER
PRINTER specifies that the output file is a line printer file.

TAPE
TAPE specifies that the output file is a tape file.

DEFAULT
The DEFAULT option can be specified for the output file only if DEFAULT is also specified for the input file. This option causes the records-per-area, number of areas, record size, and records-per-block of the output file to be identical to those of the input file. DEFAULT cannot be specified when doing a MERGE.

<areas>
Areas specifies the maximum disk areas allowed in the output file.

RELEASE
RELEASE is valid for tape files only and specifies that the tape is to be closed with release.

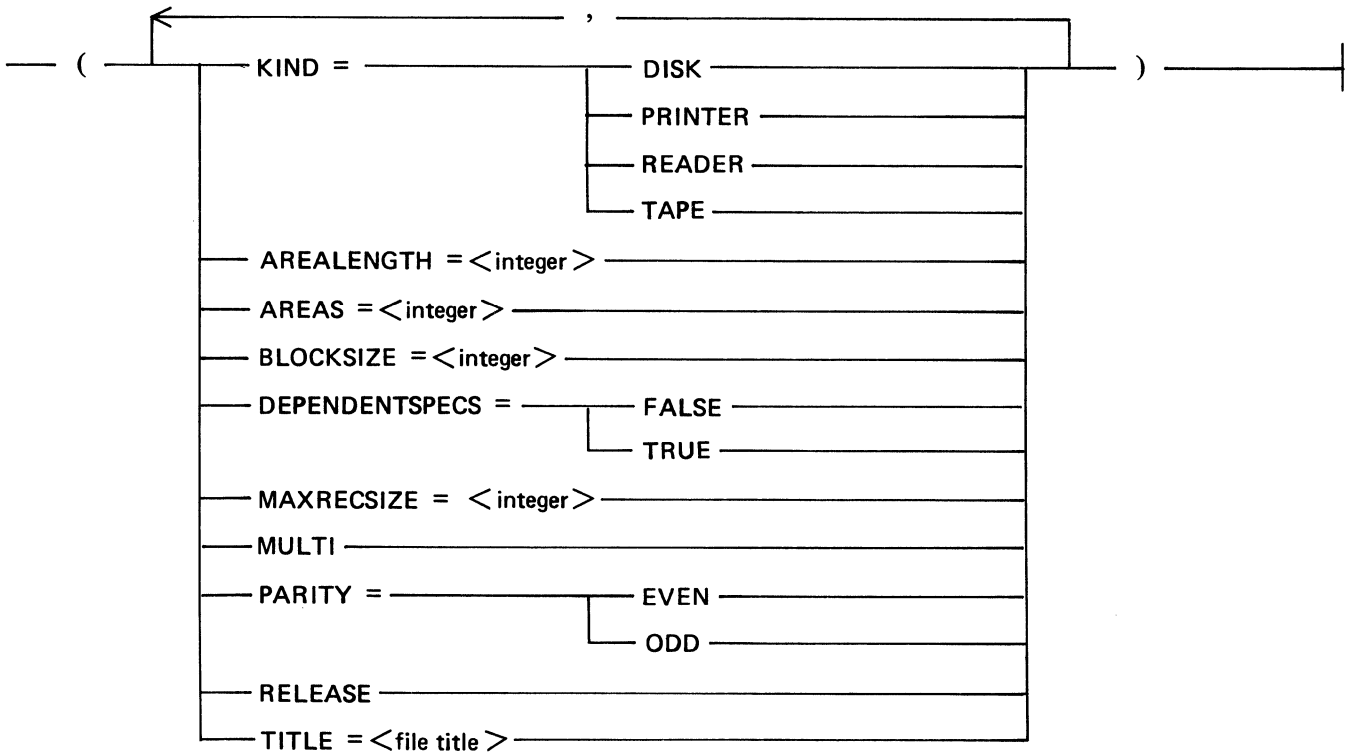
Example:

```
FILE IN DG (TAPE 80 1)
    OUT X/Y/Z (DISK (900 25) 80 9)
```

Output File Description Using File Attributes

The output file description of the FILE specification, which describes the output file of a sort or merge operation, can be specified through the use of file attributes. This new file syntax was implemented in the Mark 11.0 software release and is an addition to the previously existing syntax. The new syntax, which makes extensive use of file attributes, is functionally equivalent to the previously existing syntax.

Syntax:



Semantics:

KIND
Specifies the device to be associated with the logical file. Mnemonic values are DISK, PRINTER, READER or TAPE.

AREALENGTH
Specifies the number of bytes (characters) in an area of a disk file.

AREAS
Specifies the maximum number of areas that can be allocated for a disk file.

BLOCKSIZE
Specifies the value of the length of a block in bytes (characters).

DEPENDENTSPECS
Specifies whether the attributes BLOCKSTRUCTURE, MAXRECSIZE, and BLOCKSIZE of a logical file are to be changed to agree with corresponding values of an associated permanent file. Mnemonic values are TRUE or FALSE.

MAXRECSIZE
Specifies the maximum size of records in bytes (characters).

MULTI
The MULTI option specifies that the file is a multipack disk file.

PARITY
Specifies the parity used for TAPE or PAPERTAPE files. Mnemonic values are EVEN or ODD.

RELEASE
RELEASE is valid for tape files only and specifies that the tape is to be closed with release.

TITLE
Specifies the external <file title> in the form B/C ON A. The B/C porton represents the <first name> and the <second name> of the file. The A portion represents the pack <family name>. Valid titles are:

B
B/C
B ON A
B/C ON A

Pragmatics:

1. The TITLE attribute is optional for all files, except DISK files.
2. The KIND attribute is required.
3. The PARITY attribute is for TAPE files only, but is optional. .
4. For DISK files, either the DEPENDENTSPECS attribute may be TRUE or the AREALENGTH attribute can be set. DEPENDENTSPECS is equivalent to DEFAULT in the old file syntax.
5. AREALENGTH is equivalent to records per area times record size in the old file syntax.
6. The AREAS attribute is for DISK files only, but is optional. AREAS is not allowed if DEPENDENTSPECS = TRUE.
7. The MAXRECSIZE attribute is required, except when KIND = DISK and DEPENDENTSPECS = TRUE. MAXRECSIZE is equivalent to record size in the old file syntax.
8. The BLOCKSIZE attribute is optional but is not allowed if DEPENDENTSPECS = TRUE. BLOCKSIZE is equivalent to records per block times record size in the old file syntax.
9. The RELEASE attribute is optional. It is not allowed for DISK files.
10. The MULTI attribute is for DISK files only and is optional.

Example:

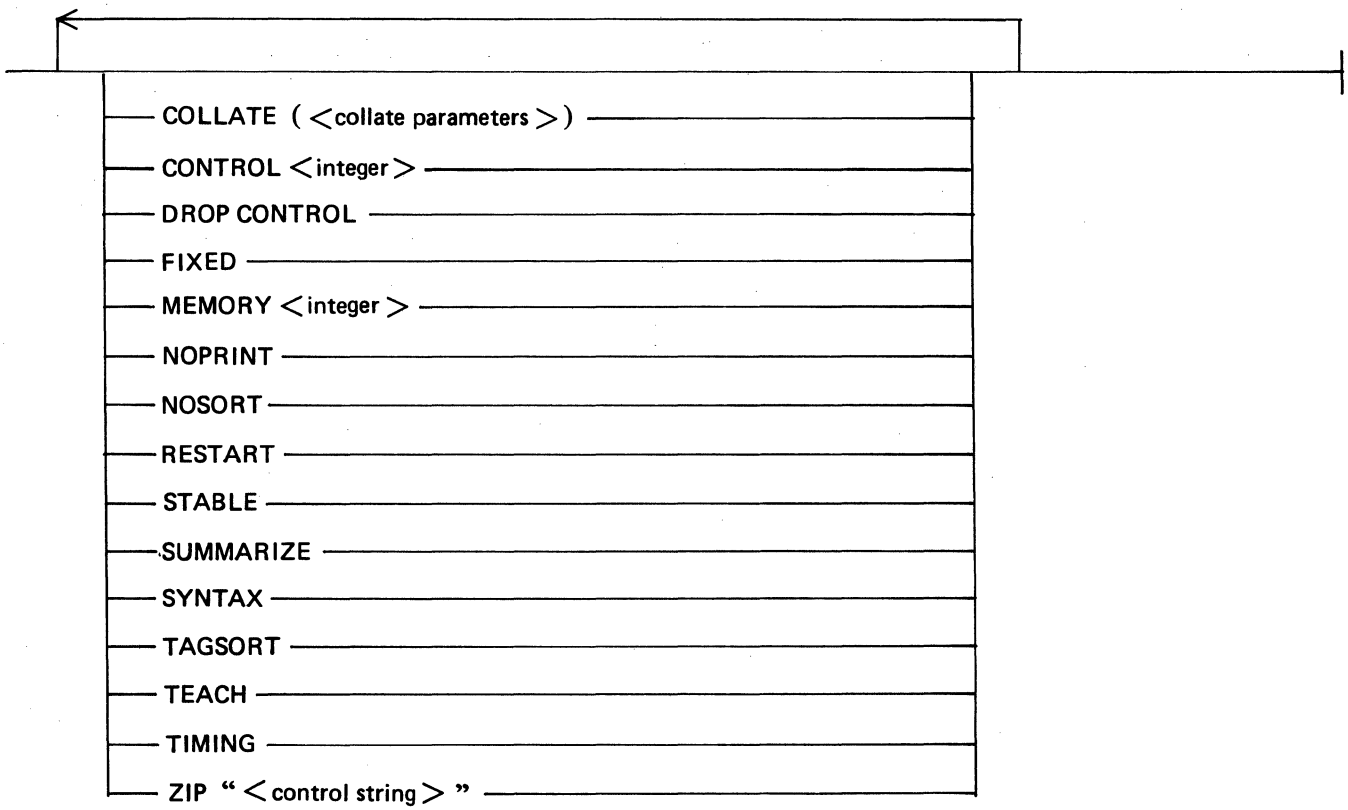
```
FILE IN (TITLE = ONE,
         KIND = DISK,
         DEPENDENTSPECS = TRUE)
      IN (TITLE = TWO,
         KIND = DISK,
         DEPENDENTSPECS = TRUE)
      OUT (TITLE = THREE,
         KIND = DISK,
         AREALENGTH = 1440,
         AREAS = 1,
         MAXRECSIZE = 80,
         BLOCKSIZE = 160)
KEY (1 5 A UA)
```

Option Specifications

The SORT/UTILITY program option specifications have the following functions:

1. Controlling the configuration or operation of the sort.
2. Causing optional operations to be performed before, during, or after the sort execution.

Syntax:

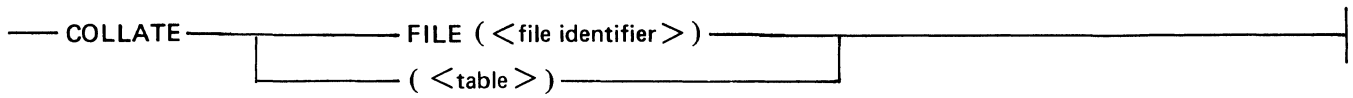


Each of the SORT/UTILITY program option specifications is described in detail in the following paragraphs.

COLLATE Option

The COLLATE option invokes an optional, virtual collating-sequence capability according to a user-specified collate file. The collate file can be used to specify a new or unique collating sequence, or to retain the standard collating sequence with the exception of certain characters being interchanged or made equal in rank. Alteration of the collating sequence may be desired for foreign alphabets or to convert from one processing system to another.

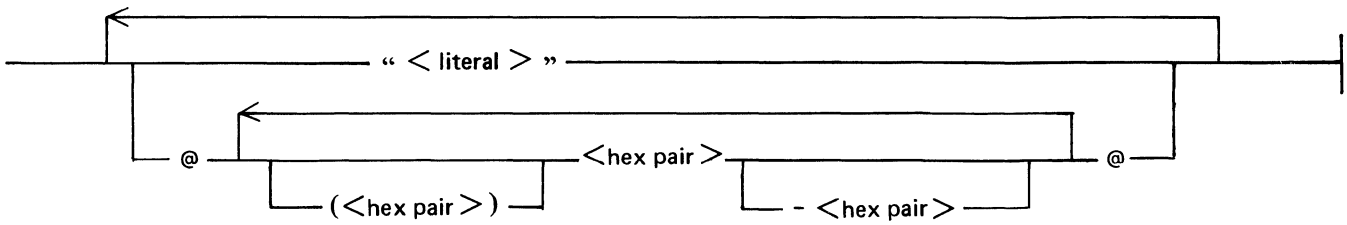
Syntax:



G50268

The <file identifier> is the name of the collate file.

The syntax of the <table> is as follows:



G50269

The <table> specifies the collate table and consists of alphanumeric literals enclosed in quotation marks and hexadecimal strings. Hexadecimal strings can be specified as a range, using the hyphen as the range specifier (for example, 00-FF). The first literal or hex (hexadecimal) pair collates as 00, the second one as 01, and so forth. If a hex pair or hex string is preceded by a hex pair in parentheses, then the first hex pair collates as the hex pair in parentheses, and the next hex pair or literal collates as the hex pair in parentheses plus 1, and so forth.

If a <file identifier> is specified, the <table> is assumed to exist on disk with the name specified.

Examples:

```
COLLATE FILE (COLLATE/FILE)
COLLATE(@00-C0@"ABCDEFGHI"@CA-D0@
        "JKLMNOPQR"@DA-E1@"STUVWXYZ"@EA-EF@@F0-FF@)
COLLATE(@(F0)40@)
```

In the first example, the collate file COLLATE/FILE is assumed to exist on disk. For clarity, the collating sequence in the second example is the default collating sequence to be used if the collate option is not specified in the SORT/UTILITY program specifications. The third example indicates that blanks (@40@) are to collate as zeroes (@F0@). In this example, both blanks and zeroes collate as zero (@F0@).

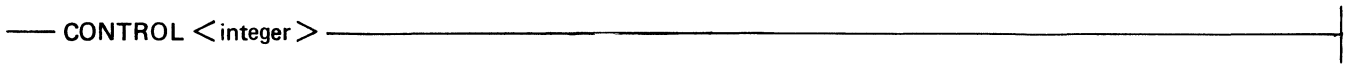
If the specified table contains less than 256 entries, the unspecified entries collate as themselves.

Signed key fields are not to be specified when collating.

CONTROL Option

The CONTROL option specifies the maximum length in bytes of the control field. If CONTROL is not specified, the default is the maximum length declared in the record specifications. Refer to the section titled Record Specifications for details on control fields.

Syntax:



G50271

Example:

```
CONTROL 10
```

DROP CONTROL Option

Specifying DROP CONTROL causes SORT/UTILITY to drop the control field from the output record. Refer to the section on Record Specifications for details.

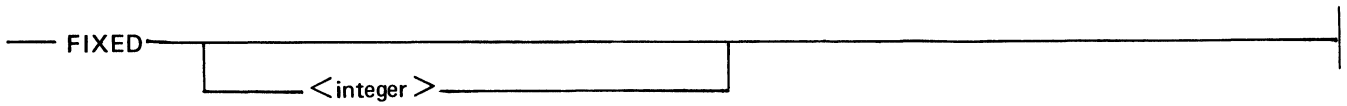
Example:

```
DROP CONTROL
```

FIXED Option

Specifying FIXED with no integer following causes the SORT/UTILITY program to expect fixed format input for the remainder of the specifications. Specifying FIXED followed by an integer causes the SORT/UTILITY program to expect the number of cards specified by <integer> to be in the fixed format. The FIXED option must be the only specification on that card.

Syntax:



G50272

Examples:

```
FIXED
FIXED 5
```

MEMORY Option

The MEMORY or ME option is used to allocate more memory to the SORT/VSORT intrinsic than the 8000 bytes assigned by default. The number of bytes must follow the word MEMORY or ME and be within the range of 8000 to 125000.

Increasing the memory available to the sort is the most significant means of increasing sort efficiency to the optimum sort memory size. Increasing the memory size beyond the optimum size does not cause the sort to run faster and may actually cause it to run slower. The optimum memory size varies according to file size (record size, blocking factor, and number of records) and other jobs in the mix at the time of the sort. To determine the optimum size for any given file, the user is advised to experiment with different memory sizes.

Examples:

```
MEMORY 12000
ME 24000
```

NOPRINT Option

The NOPRINT option is used to inhibit the printing of the SORT/UTILITY specifications on the line printer, and if used, must be the first entry in the SORT/UTILITY specification deck.

Example:

```
NOPRINT
```

NOSORT Option

The NOSORT option creates the output file as specified, but no sorting is done.

Example:

```
NOSORT
```

RESTART Option

A sort which is abnormally terminated may be restarted at a point following the latest successful sort pass in the following manner:

1. RESTART must be specified in the set of sort specifications for the ensuing sort to to be considered restartable.
2. After abnormal termination, the sort may be restarted by inserting a RESTART <job number> statement into the sort specification deck and re-executing the sort. The <job number> is the job number of the SORT/UTILITY program that was abnormally terminated.

Examples:

```
RESTART
RESTART 1259
```

STABLE Option

The STABLE option causes records which are equal, that is, records which have duplicate key fields, to be placed into the sorted output file in the same order in which they occurred in the input file. If STABLE is not specified, the order of such duplicate records in the output file cannot be predicted.

During the input phase an extra implicit key is appended to the keys to be used as the "tie-breaker" for records that have equal keys. During the output phase this implicit key (the input file relative record number) is removed.

Example:

```
STABLE
```

SUMMARIZE Option

Specifying SUMMARIZE causes records with duplicate control fields to have the SUMMARYDATA fields summed together and includes only one of the duplicate records in the output file. Both SUMMARIZE and TAGSORT must not be specified in the same set of SORT/UTILITY specifications. Refer to the SUMMARYDATA specification for details.

Example:

```
SUMMARIZE
```

SYNTAX Option

The SYNTAX option is used when the SORT/UTILITY specification cards are to be checked for errors only. No sorting is done.

Example:

```
SYNTAX
```

TAGSORT Option

The TAGSORT option provides a means of sorting a file and of creating an output file containing indices that point to the relative locations of records within the original file. The input file remains intact.

Input files can originate from any allowable hardware device type, but must reside on disk when the file of indices is being used to access the actual records.

The output file, referred to as the tag file, must be defined as four characters per record because each record consists of an 8-digit numeric number, and each is used as an index that points to the corresponding record in the input file.

Example:

```
TAGSORT
```

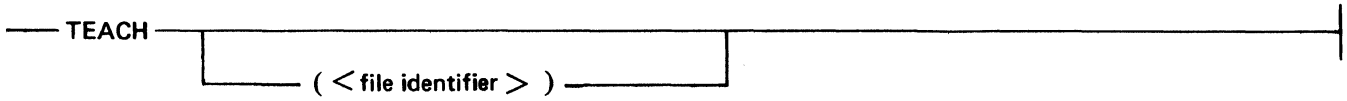
Access to the input file is provided with the tag file as follows:

1. When sorting in COBOL, the specified access method is RANDOM and the tag-file record is used as the ACTUAL KEY.
2. When sorting in RPG, the specified access method is INDEXED and the relative record number is used as delivered in the tag file to directly (DIRECT) access the original file.

TEACH Option

The specifications to the SORT/UTILITY program can be in two formats: free-form format and fixed format. If fixed format is used, then the TEACH command causes the SORT/UTILITY program to print the equivalent free-form format specifications.

Syntax:



G50270

If TEACH is followed by a <file identifier>, then a disk file which contains the free-form format specifications is created with the name specified by the <file identifier>.

Examples:

TEACH
TEACH (NEW/SORTSPECS)

TIMING Option

At the end of the sort execution, if TIMING is specified, the SORT/UTILITY program displays on the operator display terminal (ODT) the number of records sorted, the elapsed time (ET) for the sort, the processor time (PT) used by the SORT/UTILITY program, and the dynamic memory used by the SORT/UTILITY program.

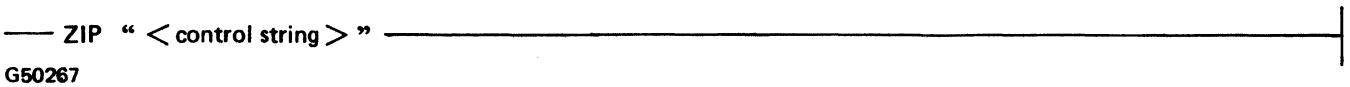
Example:

```
TIMING
```

ZIP Option

The ZIP option allows a control string to be passed (zipped) to the MCP before the sort program goes to end of job.

Syntax:



G50267

The <control string> must be delimited by quotation marks. If a quotation mark is to be included in the zip string, use two consecutive quotation marks ("").

Only one ZIP specification, limited to a maximum length of 255 characters can be contained in the SORT/UTILITY specifications.

Examples:

```
ZIP "EX DMPALL"
ZIP "COPY AND COMPARE ""(USERA)""/= TO TAPEFILE"
```

Record Specifications

Different input and output record formats may be specified by the INCLUDE or DELETE specification, followed by the output record format descriptions. Different record formats are described using subsequent INCLUDE specifications and output record format descriptions.

If the INCLUDE or DELETE specifications are to be used, then an INCLUDE or DELETE specification must precede any output record format descriptions.

If DELETE specifications are used, then they must be immediately followed by an INCLUDE specification.

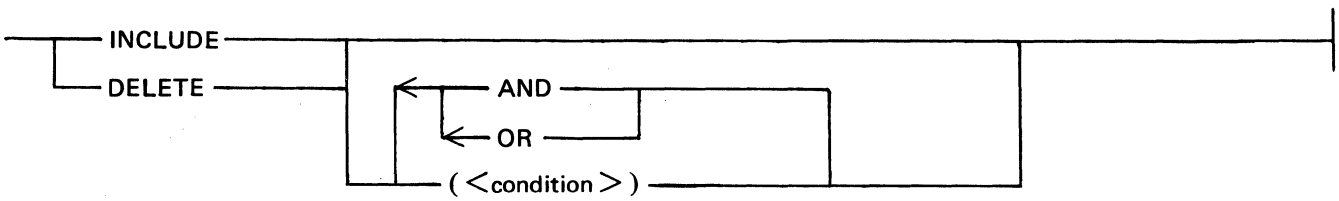
If a DELETE specification is used and if the rest of the input records are to be included, then specify INCLUDE with no parameters. Also, if the remaining input records are to be included, the last entry must be an INCLUDE with no parameters. Once INCLUDE with no parameters is encountered, no other INCLUDE/DELETE specification may appear.

The output record format description follows the INCLUDE/DELETE specification. The output record format must be explicitly specified by the user. The output record format is divided into two portions: the control field and the data field. The control field is used as the key for the sort. The control field, which must be specified before the data field, appears in the output record before the data field. The maximum length of the control field may be specified in the CONTROL option. The control field is dropped in the final output record if the DROP CONTROL option is specified. The key fields can be respecified in the data field (for position sake) by the data specification, but the key fields in the data field are not manipulated by the sort. The control field is specified by using a series of KEY and FORCEKEY specifications. The data field is specified by using a series of DATA, FORCEDATA, SUMMARYDATA, and OVERFLOW specifications. Each field in the output record is specified sequentially: first, using a series of KEY and FORCEKEY specifications; second, by using a series of DATA, FORCEDATA, SUMMARYDATA, and OVERFLOW specifications.

INCLUDE/DELETE Specification

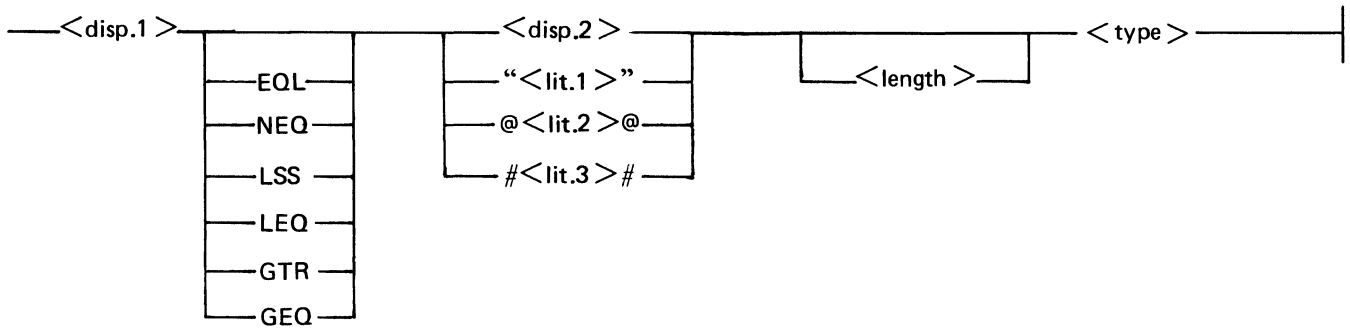
The INCLUDE/DELETE specification allows the inclusion or exclusion of certain records by comparing relational evaluations of portions of the record with other portions of the same record or with specified literals. When collating, all comparisons are made in the collated state.

Syntax:



G50273

The syntax of the <condition> is as follows:



G50274

The reserved words INCLUDE and DELETE can be followed by more than one key. Multiple keys within an INCLUDE or DELETE specification are joined by the reserved words AND or OR. The resulting expressions are evaluated element by element, left to right. If previous expressions have been evaluated as true when an OR is encountered, the evaluation is terminated as true. If previous expressions have been evaluated as false when an AND is encountered, the expressions that follow are bypassed until an OR is encountered. An example of the evaluation, where parentheses illustrate precedence and where A, B, C, and D are conditions, follows:

Example:

(A AND B) OR (C AND D)

<Disp.1> specifies the offset of the key (first operand) from the beginning of the input record. The offset is in bytes or digits, depending upon the type entry.

The entry following the key displacement is the relational operator to be applied to the comparison of the specified operands. Valid entries are EQL (equal), NEQ (not equal), GTR (greater), GEQ (greater or equal), LSS (less), and LEQ (less than or equal). The default, if no entry is specified, is equal.

The second operand (disp.2) may be specified as a displacement into the record or as a literal. As a displacement, the second operand is subject to the rules given for the first operand. Hexadecimal literals are delimited by at signs (@). Character literals are delimited by quote marks. Numeric literals are delimited by pound signs (#). All characters are valid except the quote and question mark. Literals are padded or truncated as necessary, based on the length attribute. The literal must be contained on one card and must not cross card boundaries.

The <length> field specifies the length of the operands to be compared. This length is 8-bit or 4-bit units depending on the type entry. The default, if no length is specified, is one.

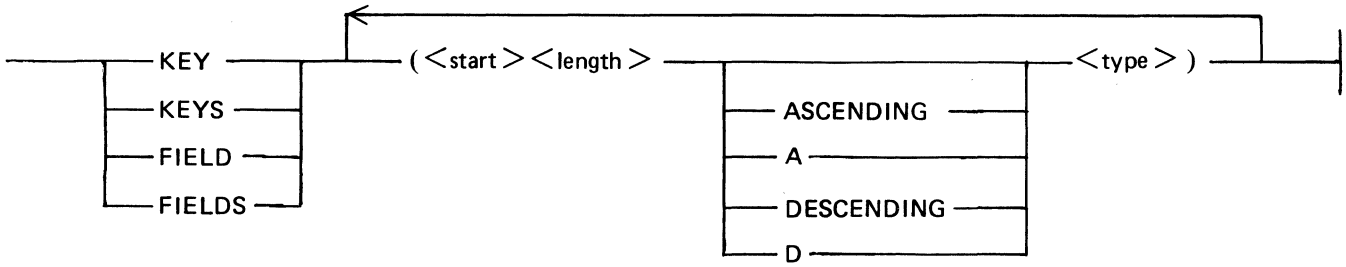
The last entry (type) specifies the type of operands and shows whether the operands are character or digit oriented. The syntax and meaning of the type is the same as in the key specification. Refer to the key specification for details.

Examples:

```
DELETE (1 EQL "ABCD" 4) AND (10 NEQ #-123# 4 SN)
INCLUDE
```

KEY Specification

A series of KEY and FORCEKEY specifications are used to specify the output record control field. The KEY specification has the following syntax:



G50275

Semantics:

<start>
This must be an integer that specifies the position of the most significant byte or digit (alphanumeric or numeric) of the field including the sign, if any, relative to the first byte or digit of the input record. The first byte or digit of a record is considered to be relative position one. The position is referred to in the number of units applicable to the type of the key.

<length>
The length must be an integer that specifies the number of bytes or digits in the key and includes the sign where applicable. The length of a key cannot exceed 511 bytes or 1023 digits.

ASCENDING or A
The key is ordered in ascending order and the lowest key value appears first. ASCENDING is the default if ASCENDING or DESCENDING is not specified.

DESCENDING or D
Specifying DESCENDING or D results in the key being ordered in descending order and the highest key value appears first.

ALPHA or UA
ALPHA or UA (unsigned alphanumeric) indicates that the data is alphanumeric, and the key location in the input record is counted in 8-bit units from the beginning of the record. Alphanumeric is the default if no data type is specified.

DIGIT
DIGIT specifies that the key is 8-bit alphanumeric and is to be sorted on the last four bits of every byte in the key.

NUMERIC or UN
NUMERIC or UN (unsigned numeric) indicates that the data is 4-bit numeric, and the relative position of the field is counted in 4-bit units.

RSA
RSA (right-signed alphanumeric) means that the data is signed 8-bit alphanumeric, and the minus sign is represented by a hexadecimal D and is located in the first four bits of the rightmost byte of the key.

RSN
RSN (right-signed numeric) means that the data is signed 4-bit numeric, and the minus sign is represented by a hexadecimal D in the rightmost digit of the key.

SA
SA (signed alphanumeric) indicates that the data is 8-bit alphanumeric and that some, or all, of the keys can contain a minus sign. The minus sign is represented by a hexadecimal D in the most significant four bits of the first byte of the key.

SN
SN (signed numeric) indicates that the data is 4-bit numeric and that some, or all, of the keys can contain a minus sign. The minus sign is represented by a hexadecimal D in the first digit of the key.

ZONE
ZONE specifies that the key is 8-bit alphanumeric and is to be sorted on the first four bits of every byte in the key.

Specifying a key causes a portion of the control field to be built. The control field is the only key used by the sort. Therefore, keys that are signed or keys that are to be sorted in descending order must be converted so that the control field can be sorted properly. If signed keys or descending keys are present, then DROP CONTROL must be specified so that converted fields do not appear in the output record.

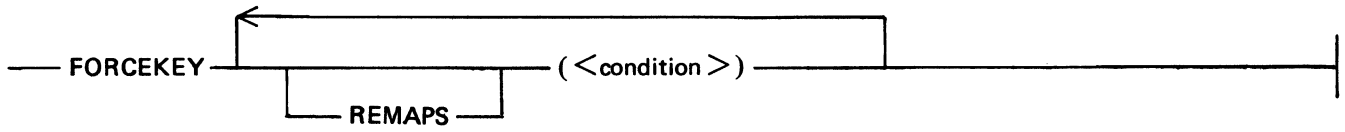
Example:

```
KEY (1 10) (20 5 SN) (30 5 D UN)
```

FORCEKEY Specification

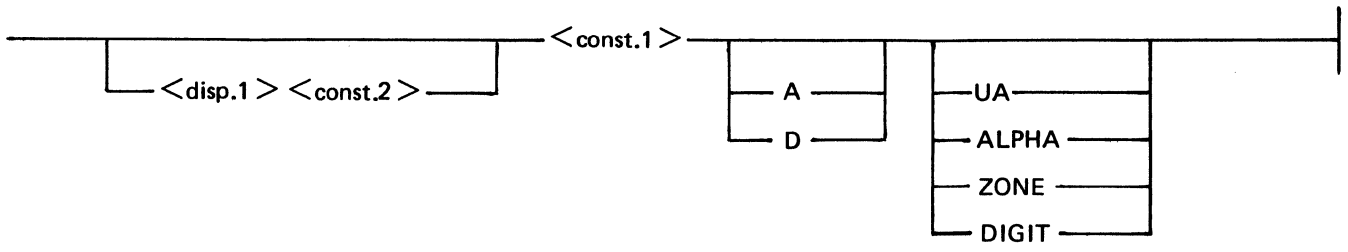
The FORCEKEY specification defines one byte of the control field of the output record.

Syntax:



G50277

The syntax of the <condition> is as follows:



G50278

<const.1> and <const.2> refer to a 1-byte constant. The constant must be an alphanumeric character enclosed in quotation marks, or two hexadecimal digits enclosed in at signs (@).

If only <const.1> is specified, then that constant is placed into the next byte of the control field of the output record.

<Disp.1> is a displacement that refers to a character in the input record at that displacement. That character is compared to the constant <const.2>. If they are equal, then the <const.1> constant is placed in the next byte of the control field of the output record. If they are not equal, then the hexadecimal value @FF@ or @00@ is placed. FF is placed if A (ascending) is specified, and 00 is placed if D (descending) is specified. @FF@ is placed if neither A nor D is specified. The data type entry specifies how the comparison of the characters is made. UA or ALPHA specifies all 8 bits of the byte, ZONE specifies the first 4 bits of the byte, and DIGIT specifies the last 4 bits of the byte.

Specifying A means that this character of the control field is to be sorted in ascending sequence. Specifying D means descending. The UA, ALPHA, ZONE, and DIGIT specifies the type of key. These two parameters have the same meaning and restrictions as in the KEY specification.

The word REMAPS indicates that this FORCEKEY specification refers to the preceding specified KEY or FORCEKEY output position. When used with a preceding FORCEKEY with a comparison, the REMAPS can be considered as an else. If the first condition is not met, the second condition is analyzed.

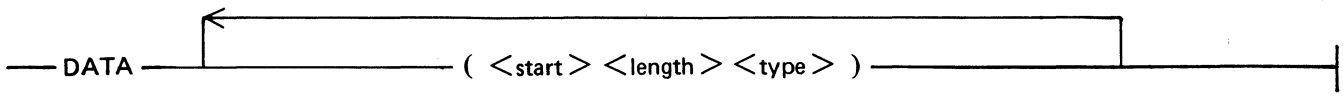
Examples:

```
FORCEKEY ("D") ("G")
FORCEKEY (1 "2" "A" A ALPHA) REMAPS ("C" A)
```

DATA Specification

A series of DATA, FORCEDATA, SUMMARYDATA, and OVERFLOW specifications are used to indicate the output record data field. The DATA specification is used to indicate that a particular portion of the input record is to appear in the output record.

Syntax:



G50279

The <start>, <length>, and <type> have the same syntax and meaning as in the KEY specification. If ZONE appears, then the output receiving field is prefilled with blanks (@40@), and only the first four bits of every byte are moved. If DIGIT appears, then the output record receiving field is prefilled with zeroes (@F0@), and only the last 4 bits of every byte are moved.

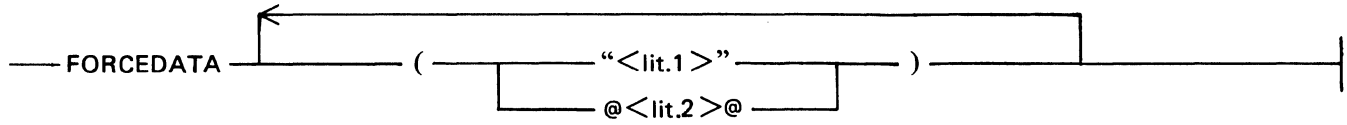
Example:

```
DATA (1 6) (7 4 ZONE) (21 2 SN)
```

FORCEDATA Specification

FORCEDATA specifies that the literal following the FORCEDATA specification is to be placed in the output record. The literal must be one alphanumeric character enclosed in quotation marks or two hexadecimal digits enclosed in at signs (@).

Syntax:



G50280

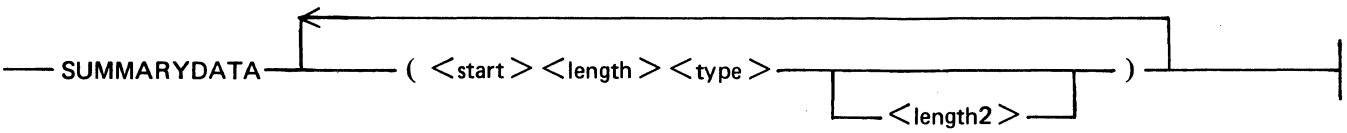
Example:

```
FORCEDATA ("D") (@C7@)
```

SUMMARYDATA Specification

The SUMMARYDATA specification is used with duplicate control fields. Those records having the same duplicate control field have the fields specified in the SUMMARYDATA specification added together, and the sum is placed in the output record. All but one of the duplicate records are removed from the output file. If STABLE is specified, then the record that is kept is the last record of the series of duplicate records.

Syntax:



G50281

The <start>, <length>, and <type> have the same syntax and meaning as in the KEY specification and refer to the field in the input record that is to be added if a duplicate record occurs. The <length2> entry is optional and specifies the length in bytes or digits of the field in the output record. Default is the length of the input field.

The SUMMARYDATA fields must be in the same relative position in the output record for all record types (all series of INCLUDE/DELETE specifications). The fields do not have to refer to the same positions in the input records. The first record type description that contains SUMMARYDATA specifications defines the summary format for all records of any type. If SUMMARYDATA is omitted or misplaced in a record specification, then a warning appears on the SORT/UTILITY output printer listing, and a SUMMARYDATA field is automatically placed in the appropriate place in the output file data field.

When ALPHA or UA fields are being summarized, a binary add of the hexadecimal values is performed.

If the SUMMARIZE option is not specified, then the SUMMARYDATA specifications are treated as DATA specifications.

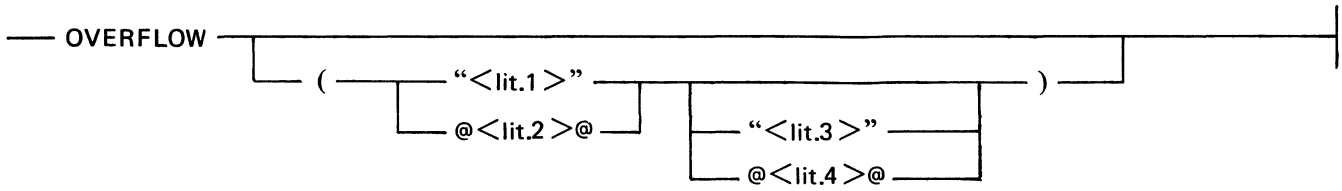
Example:

```
SUMMARYDATA (51 4 SN 8)
```

OVERFLOW Specification

The OVERFLOW specification specifies one byte of the data field and is used to indicate whether an overflow has occurred when adding any SUMMARYDATA fields.

Syntax:



G50282

The literals must be one alphanumeric character enclosed in quotation marks, or two hexadecimal digits enclosed in at signs (@). If an overflow occurs in any SUMMARYDATA field, <lit.1> or <lit.2> is placed in the output record. If no overflow occurs, then <lit.3> or <lit.4> is placed in the output record. The default character for <lit.1> or <lit.2> is an asterisk (*), and the default for <lit.3> or <lit.4> is a blank. The <lit.1>, <lit.2>, <lit.3> and <lit.4> entries must be the same for all record types.

Example:

```
OVERFLOW ("*" @40@)
```

Fixed Format Specifications

Fixed format specifications are described in this subsection. Fixed format specifications are compatible with the specifications used in the IBM $DSORT program. Fixed format specifications must be preceded by the FIXED free-form format specification.

Numeric data must be byte-aligned when using the fixed format specifications. When specifying position, the relative byte is used.

Descriptions of the equivalent free-form format specification are given where possible.

There are four types of fixed format input: header, ALTSEQ, include or delete, and field specifications. The function and order of appearance of the Include/Delete and field specifications are the same as the free-form format INCLUDE/DELETE, KEY, FORCEKEY, DATA, SUMMARYDATA, FORCEDATA, and OVERFLOW specifications.

Header Specification

The header specification is used to indicate the type of sort and some of the parameters for the sort. The format and a description of the header specification is as follows:

Columns	Function and Description
1-5	Sequence number: A 5-digit sequence number must be entered, consisting of a 2-digit page number and a 3-digit line number. If the sequence numbers are not in order, a warning message is issued.
6	Line type: This column must contain the letter H to show that it is a header line.
7-12	Type of job: The type of sort is specified in these columns. SORTR means tag-along (normal) sort. SORTA means addrout sort, which is equivalent to specifying the free-form format TAGSORT specification. SORTRS means a summary tag-along sort, which is equivalent to specifying the free-form format SUMMARIZE specification.
13-17	Control length: A right-justified number specifying the maximum length in bytes of the control field must be entered in columns 13-17. This is equivalent to the free-form format CONTROL specification. The length must not exceed 511 bytes.
18	Order: Indicates the order in which the records are to be sorted. Enter the letter A for ascending or the letter D for descending.
19	Signed Fields: This column is used to specify whether signed fields are right or left signed. The letter L in this column specifies that all signs are on the left of the field. If the column is blank, then all signs are assumed to be on the right side of the field.
20	File close: Column 20 is left blank if the input file is not to be closed with purge. Entry of the letter P indicates that the input file is to be closed with purge. When there is more than one input file, this column applies to the first input file only, and the other files are closed with release.
21-25	Blank: Columns 21-25 are not used and are to remain blank.
26	Collating sequence: Column 26 is left blank if the standard collating sequence is to be used. If an alternate collating sequence is to be used, this column must contain the letter S. If an S is specified, then the collate table must be specified by either the free-form format COLLATE specification or the fixed format ALTSEQ specification.
27	Blank: Column 27 is not used, and is to remain blank.

Columns	Function and Description
28	Control field output: The letter X in this column indicates that the control field is to be dropped from the output record, which is equivalent to the free-form format DROP CONTROL specification. Control fields are normally dropped if opposite control fields or an alternate collating sequence is used. If the control field is dropped, then the keys also should be respecified in the data field as data, so that the keys are not lost.
29-32	Output record length: Columns 29-32 must contain a right-justified number indicating the output record size in bytes. The size must not exceed 8191 bytes.
35-39	Sort additional memory: This is equivalent to the free-form format MEMORY specification. The number that is specified (right justified) is added to 8000 bytes to indicate the sort memory size. To specify the free-form format equivalent of MEMORY 12000, this field should contain 4000.
40-96	Comments: Columns 40-96 are used for comments and are ignored.

ALTSEQ Specification

The ALTSEQ specification is used to specify the alternate collating sequence. Its function is identical to the COLLATE specification. Each character is specified by using four hex (hexadecimal) digits. The first two hex digits refer to the character that is to be collated, and the second two hex digits refer to the hex value that character is to collate (translate) as. When reading the input file or files, the SORT/UTILITY program translates the keys according to the replacement values specified, before determining inclusion or deletion or doing the sort. Because of this translation, signed key fields cannot be used with an alternate collating sequence.

The ALTSEQ statements must appear immediately after the header specification and before any Include/Delete or field specifications.

The format of the ALTSEQ specification is as follows:

Columns	Function and Description
1-6	Identifier: Columns 1-6 must contain the literal ALTSEQ to identify this as an ALTSEQ specification.
7-8	Blank: Columns 7-8 are not used and must remain blank.
9-96	Collate values: Columns 9 through column 96 inclusive contain sets of 4 hexadecimal digits that refer to the collating changes. There are no embedded blanks between sets. The sets may be terminated at any point on the input specification and continued on the next input specification. Any following ALTSEQ specifications are identical in format, up to the number necessary to specify the collate changes.

The series of ALTSEQ statements must be terminated with an input consisting of two asterisks (*) in columns 1 and 2, and the remainder of the input must be blank.

Example:

```
ALTSEQ F0404ED0
ALTSEQ D1D2
**
```

Include and Delete Specification

The fixed format Include/Delete specification performs the same functions as the free-form format INCLUDE/DELETE specification. The format and a description of the fixed format Include/Delete specification follows:

Columns	Function and Description
1-5	Sequence number: A 5-digit sequence number must be entered, consisting of a 2-digit page number and a 3-digit line number. If the sequence numbers are not in order, a warning message is issued.
6	Line type: Column 6 must contain either the letter I to indicate include, or the letter O to indicate omit. To specify the equivalent of the free-form format INCLUDE ALL, columns 7 through 39 must be blank.
7	Continuation or comment: Column 7 indicates the relation of this specification to the preceding Include/Delete specification. A blank indicates that this is the first of a series of include or delete entries, or a single include or delete entry. The letter A indicates an AND relation with the previous include or delete entry. The letter O indicates an OR relation with the previous include or delete entry. An asterisk (*) indicates that this line is a comment line, and the line is to be ignored.
8	Interpretation: Specifies whether the conditions for include or delete should be compared as characters or numeric data. The letter C indicates unsigned alphanumeric characters (UA in free-form format). The letter Z indicates the zone portion of the characters (ZONE in free-form format). The letter D indicates the digit portion of the characters (DIGIT in free-form format). The letter P indicates signed packed numeric data (SN in free-form format). The letter U means signed unpacked alphanumeric characters (SA in free-form format). The letter P or the letter U must not be specified if an alternate collating sequence is used (COLLATE or ALTSEQ).
9-12	Compare Field Start: Columns 9-12 must contain a right-justified number indicating the starting location byte in the input record of the field to be compared for determination of inclusion or deletion. If the field is one byte in length, these columns must remain blank. The first byte of the input record is counted as position 1.
13-16	Compare Field Stop: Columns 13-16 must contain a right-justified number to indicate the stopping location byte in the input record of the field to be compared for determination of inclusion or deletion. If the field is one byte in length, these columns must contain the location of the field.

Columns	Function and Description
17-18	Relation: Columns 17-18 indicate the relation of the fields to be compared for determination of inclusion or deletion. The following entries can be used:

Entry	Relation
EQ	Equal to
NE	Not equal to
LT	Less than
LE	Less than or equal to
GT	Greater than
GE	Greater than or equal to

19 Field or Constant: Column 19 indicates that the field in the input record is to be compared to a literal or to another field in the input record. The letter C indicates a constant, and the letter F indicates another field in the record.

20-39 Compare Field: Columns 20-39 contain the literal or the second field location of the value with which it is to be compared. If a literal is used, then columns 20 through 39 contain the literal. If the literal is alphanumeric, the data is left justified. If the literal is numeric, the data is right justified. If a second field in the input file is used, columns 20 through 23 contain the starting location of the field in the input record, and columns 24 through 27 contain the stopping location of the field in the input record. For fields of length one, columns 20 through 23 are to remain blank and columns 24 through 27 are to indicate the position. The literal or field must be the same length as the value with which it is compared.
When coding signed packed (numeric) literals, the leftmost or rightmost character, depending on whether the fields are left-signed or right-signed, must contain its sign (+ or −). When coding negative signed unpacked (alpha) literals, the leftmost or rightmost character (number) must be coded as follows:

Number	Replace By	Hex Equivalent
0	}	@D0@
1	J	@D1@
2	K	@D2@
3	L	@D3@
4	M	@D4@
5	N	@D5@
6	O	@D6@
7	P	@D7@
8	Q	@D8@
9	R	@D9@

40-96 Comments: Columns 40-96 are used for comments and are ignored.

Field Specification

The field specification is used to define the control and data fields. The equivalent free-form format specifications are KEY, FORCEKEY, DATA, FORCEDATA, SUMMARYDATA, and OVERFLOW. The order of the specifications is the same as in the equivalent free-form format specifications. The format and a description of the fixed format field specification is as follows:

Columns	Function and Description
1-5	Sequence number: A 5-digit sequence number must be entered, consisting of a 2-digit page number and a 3-digit line number. If the sequence numbers are not in order, a warning message is listed on the SORT/UTILITY listing.
6	Line Type: Column 6 must contain the letter F to show that this is a field description line.
7	Field Type: Column 7 indicates which field type is being defined. The letter N indicates a normal key field, and the letter O indicates an opposite key field. Both N and O represent the same thing as in the free-form format KEY specification, with one difference. If column 18 of the header specification contains the letter A (ascending), then specifying an N indicates that the key is to be sorted ascending. Specifying an O indicates that the key is to be sorted descending. If, however, column 18 of the header specification contains the letter D (descending), then specifying an N indicates that the key is to be sorted descending, and specifying an O indicates that the key is to be sorted ascending. The letter F in column 7 indicates a forced control field equivalent to the free-form format FORCEKEY specification. The letter D indicates a data field equivalent to the free-form format DATA specification. The letter S indicates a summary data field equivalent to the free-form format SUMMARYDATA specification. An asterisk indicates that the line is a comment, and the line is ignored.

Column 7 is summarized below:

Column 7	Meaning
N	Normal control field
O	Opposite control field
F	Forcekey specification
D	Data specification
S	Summarydata specification
*	Comment line

Columns	Function and Description
8	Data Type: Column 8 defines the type of data. The letter C indicates unsigned alphanumeric characters equivalent to the UA free-form format specification. The letter Z indicates the zone portions of the characters equivalent to the ZONE free-form format specification. The letter D indicates the digit portions of the characters equivalent to the DIGIT free-form format specification. The letter P indicates signed packed numeric data equivalent to the SN free-form format specification. The letter U indicates signed unpacked data equivalent to the SA free-form format specification. A U or P must not be specified if an alternate collating sequence, that is COLLATE or ALTSEQ, is used. If column 7 contains the letter D (data field), then column 8 may contain the letter V, which indicates that a character is to be forced into the data field equivalent to the FORCEDATA free-form format specification. The entry choices for column 8 follow:
9-12	Field Location Start: Columns 9-12 contain a right-justified number indicating the starting location (byte) of the field in the input record. If the field is of length 1, then these columns are to remain blank. The first byte of the input record is considered position 1.
13-16	Field Location Ending: Columns 13-16 contain a right-justified number indicating the ending location (byte) of the field in the input record. If the field is of length 1, then these columns contain the location of the field.
17	Compare Character: Column 17 is used only when column 7 contains the letter F(forcekey specification) and the force is only to be done if the character indicated in columns 13 through 16 is equal to the character placed in this column. If this column is blank, then the comparison is not made, and the character in column 18 is unconditionally forced.
18	Forced Character: Column 18 contains the character to be forced, which is used when the letter F is in column 7 (Forcekey) or the letter D in column 7 and the letter V in column 8 (Forcedata).

Column 8	Meaning
C	Unsigned alphanumeric (UA)
Z	Zone portion (ZONE)
D	Digit portion (DIGIT)
P	Signed numeric (SN)
U	Signed alphanumeric (SA)
V	Forcedata specification

Columns	Function and Description
19	Remaps: Column 19 is used when column 7 contains the letter F (Forcekey) and this forcekey remaps the previous forcekey. A blank indicates that this forcekey does not remap the previous one, and a non-blank character indicates that this forcekey does remap a previous one. Refer to the free-form format FORCEKEY specification for details on REMAP.
20-22	Overflow Field Length Entry: Columns 20-29 contain the length of the overflow field. When doing a summarize sort, and the sum is larger than the size of the field in the input record, then the field is used to specify a larger field in the output record. These columns contain a right-justified number indicating the length of the output field in bytes.
23-39	Blank: Columns 23-29 are not used and are to remain blank.
40-96	Comments: Columns 40-96 are available for comments and are ignored.

SORT/UTILITY Program Example Using Fixed Format

The fixed format specifications are shown in the following example. The TEACH option was used to direct the compiler to print out the equivalent free-form format specifications.

```
FILE IN TODAY (DISK DEFAULT)
    OUT YESTERDAY (DISK DEFAULT)
TEACH
FIXED
00100HSORTA 8A
00200I C 17 17NEC
00300IAC 2 3EQCIO
00400IAC 5 5EQC5
00500IOC 17 17NEC
00600IAC 2 3EQC60
00700IAC 5 5EQC5
00800FNC 2 4
00900FNC 17 17
01000FNC 12 15

   * * * TEACH FILE * * * "TEACH"

TAGSORT
INCLUDE (17 NEQ @40@ 1 UA)
AND (2 EQL @C9D6@ 2 UA)
AND (5 EQL @F5@ 1 UA)
INCLUDE (17 NEQ @40@ 1 UA)
AND (2 EQL @F6F0@ 2 UA)
AND (5 EQL @F5@ 1 UA)
KEY (2 3 A UA)
KEY (17 1 A UA)
KEY (12 4 A UA)
```

Appendix A
SORT INTRINSIC TABLES

This appendix contains information about the tables that programs must build before a sort intrinsic is called. These tables are built automatically by the COBOL '68, COBOL '74 and RPG compilers and in the SORT and SORT/UTILITY programs. These tables are transparent to the user. In the SDL or UPL languages these tables must be set up explictly by the program. The tables in this appendix are described using the SDL/UPL language syntax.

The sort intrinsic is called by a special MCP communicate (discussed in the subsection labeled Sort Communicate). Basically, the communicate passes program base-relative addresses of the various tables to the MCP.

Sort Information Table

The Sort Information Table contains basic information and sort specifications. The format of this table follows:

```
DECLARE 01 SORT_INFORMATION_TABLE,
          02 SORT_TYPE                  BIT(2),
          02 SORT_HDWR                  BIT(6),
          02 FILLER                     BIT(8),
          02 SORT_PURGE_FILES           BIT(8),
          02 SORT_FILES                 BIT(8),
          02 SORT_RECSIZE               BIT(24),
          02 SORT_IN_HDWR               BIT(6),
          02 SORT_IN_RECSIZE            BIT(24),
          02 SORT_IN_BLKSIZE            BIT(24),
          02 SORT_IN_CLOSE              BIT(12),
          02 SORT_IN_VARIABLE           BIT(1),
          02 SORT_OUT_HDWR              BIT(6),
          02 SORT_OUT_RECSIZE           BIT(24),
          02 SORT_OUT_BLKSIZE           BIT(24),
          02 SORT_OUT_CLOSE             BIT(12),
          02 SORT_OUT_VARIABLE          BIT(1),
          02 SORT_DELETING              BIT(1),
          02 SORT_STABILIZE             BIT(1),
          02 SORT_PARITY                BIT(1),
          02 SORT_RESTART               BIT(1),
          02 SORT_BIAS                  BIT(7),
          02 SORT_RECORDS               BIT(24),
          02 SORT_REPORT                BIT(1),
          02 SORT_NUMBER_KEYS           BIT(5),
          02 SORT_TIME_IT               BIT(1),
          02 SORT_IN_OVERRIDE           BIT(1),
          02 FILLER                     BIT(6),
          02 SORT_KEY_LENGTH            BIT(16),
          02 SORT_COLLATE_KEY_LENGTH    BIT(16),
          02 SORT_PARTITION             BIT(24),
          02 SORT_DELETE_KEYS           BIT(4),
          02 SORT_DUPCHECK              BIT(1),
          02 SORT_TAGRPG                BIT(1),
          02 SORT_W1_PID                BIT(1),
          02 SORT_W2_PID                BIT(1),
          02 SORT_TAGCOBOL              BIT(1),
          02 FILLER                     BIT(4),
          02 SORT_ISAM_FILE_IN          BIT(1),
          02 FILLER                     BIT(10),
          02 SORT_MEMORY                BIT(24),
          02 SORT_TAGSEARCH             BIT(1),
          02 SORT_COLLATE               BIT(1),
          02 FILLER                     BIT(31),
          02 SORT_RESTART_JOB           BIT(24);
```

The meaning and possible values of the variables in the table follow:

SORT_TYPE
Indicates which intrinsic is to be called. 0 means SORT/VORT, a 1 means SORT/QSORT, 2 means a tagsort with SORT/VORT, and 3 means SORT/MERGE.

SORT_HDWR
Indicates which hardware type is to be used for the work files. If the value is 17, SORT/VSORT is invoked using disk as the work file storage media. If the value is 27, SORT/TAPESORT is invoked with work files on magnetic tape.

SORT_PURGE_FILES
Indicates which input files are to be purged.

SORT_FILES
If SORT/TAPESORT is to be invoked, this field indicates how many worktapes to use. This value must be greater than 3 and less than 9.

SORT_RECSIZE
The maximum record size in bits.

SORT_IN_HDWR
The hardware media on which the input file resides. It is coded the same way as the hardware type in File Parameter Blocks. A list of more common values follows:

Hardware Type	Value
Card Reader	5
Card Punch	2
Paper Tape Reader	7
Paper Tape Punch	20
Printer	8
Disk	17
Tape	27

SORT_IN_RECSIZE
The record size of the input file in bits.

SORT_IN_BLKSIZE
The maximum block size in bits of the input file.

SORT_IN_CLOSE
Indicates the manner in which the input file is to be closed. @200@ means close with purge, and @400@ means closed with release.

SORT_IN_VARIABLE
If this bit is set, it indicates that the input file has variable length records.

SORT_OUT_HDWR
Indicates the hardware media on which the output file resides. Use the same values as SORT_IN_HDWR.

SORT_OUT_RECSIZE
Indicates the output file record size in bits.

SORT_OUT_BLKSIZE
Indicates the maximum block size of the output file in bits.

SORT_OUT_CLOSE
Indicates the manner in which the output file is to be closed. @010@ means close with lock, and @400@ means close with release.

SORT_OUT_VARIABLE
If this bit is set, it indicates that the output file is to have variable length records.

SORT_DELETING
If set, indicates that there are INCLUDE/DELETE keys.

SORT_STABILIZE
If set, indicates that records that have the same key values (duplicate records) are to have the same order in the output file.in

SORT_PARITY
If set, input records that have parity errors are to be discarded.

SORT_RESTART
If set, indicates that the sort is to be restartable. SORT/VSORT saves restart information for re-starting an abnormally terminated sort.nated

SORT_BIAS
The values range from 0-99 inclusive and indicate the bias of the input file.

SORT_RECORDS
Indicates the number of records in the input file. This field indicates the number of records to be extracted and sorted if the sort is to be partitioned.

SORT_REPORT
If this bit is set, a report is printed indicating the sort parameters specified.

SORT_NUMBER_KEYS
Indicates the number of keys.

SORT_TIME_IT
If set, the elapsed sort time is displayed on the console printer after the sort is completed.

SORT_IN_OVERRIDE
If set, the input file attributes are to be obtained from the disk file header.

SORT_KEY_LENGTH
The sum of the total length of keys in bits.

SORT_COLLATE_KEY_LENGTH
The sum of the total length in bits of the keys that are to be collated.

SORT_PARTITION
If non-zero, indicates that a partition sort is to be done. It points to the starting record to be sorted, and the SORT_RECORDS field indicates how many records to sort. The first record in the file is considered record number one.

SORT_DELETE_KEYS
Indicates how many INCLUDE and DELETE keys there are.

SORT_DUPCHECK
If this bit is set, a disk file is built with pointers to the records that have duplicate keys (see DUPCHECK with no parameters in section 2).

SORT_TAGRPG
If this bit is set, a sorted tag file is built in the RPG format (see TAGRPG in section 2).

SORT_W1_PID
If set, indicates that the sort work file 1 is to be on a user pack.

SORT_W2_PID
If set, indicates that the sort work file 2 is to be on a user pack.

SORT_TAGCOBOL
If this bit is set, a sorted tag file is built in the COBOL format (see TAGCOBOL in section 2).

SORT_ISAM_FILE_IN
If set, indicates that the input file is an ISAM data file.

SORT_MEMORY
It indicates, in bits, how much sort memory to use.

SORT_TAGSEARCH
If set, a TAGSEARCH sort is done (see TAGSEARCH in section 2).

SORT_COLLATE
If set, a collate file is to be used with the input and ouput files.

SORT_RESTART_JOB
If set at non-zero, this variable indicates that a sort is to be restarted. The value is the restart job number in binary.

Sort Key Table

The Sort Key Table contains the descriptions of the keys that the operation is to act upon. The Sort Key Table has two formats, one for the key sign and one for the key. If the key is unsigned, it does have an entry for the key sign. If the key is signed, the entry for the sign immediately precedes the entry for the key. The table has space for 30 entries, with the 31st entry a null (zero) entry. If the table has less than 30 entries, the last entry is followed by a null (zero) entry. The format of the Sort Key Table follows:

```
DECLARE 01 SORT_KEY_TABLE          BIT(1116),
           02 KEY (30)             BIT(36);
```

The two formats of the key entry follow:

```
DECLARE 01 KEY_SIGN                BIT(36),
           02 SIGN_FLAG            BIT(1),
           02 FILLER               BIT(1),
           02 NEW_FORMAT           BIT(1),
           02 FILLER               BIT(1),
           02 LENGTH_AND_SIGN      BIT(12),
              03 SIGN_LENGTH       BIT(4),
              03 SIGN_CHAR         BIT(8),
           02 SIGN_DISPLACEMENT    BIT(20);
DECLARE 01 KEY_FIELD               BIT(36),
           02 SIGN_FLAG            BIT(1),
           02 DIRECTION            BIT(1),
           02 FILLER               BIT(1),
           02 COLLATE_KEY          BIT(1),
           02 KEY_LENGTH           BIT(12),
           02 KEY_DISPLACEMENT     BIT(20);
```

The meaning and possible values of the variables in the table follow:

SIGN_FLAG
Indicates which type of key this key is. If 1, this key is a sign key. If 0, this is the actual key.

NEW_FORMAT
If this bit is reset, the LENGTH_AND_SIGN field is automatically changed to @40D@. If this bit is set, the field is unchanged (refer to LENGTH_AND_SIGN).

LENGTH_AND_SIGN
These fields contain the length of the sign character in bits and the actual minus sign character.

SIGN_LENGTH
Contains the length of the sign character. Normally, this value is set to 4 (4 bits).

SIGN_CHAR
Contains the actual minus sign character. Normally, this value is set to @0D@.

SIGN_DISPLACEMENT
Indicates the location of the sign in bits from the beginning of the record. The first bit of the record is considered relative bit 0.

DIRECTION
Indicates whether this key is to be sorted in ascending or descending order. 0 means ascending, and 1 means descending.

COLLATE_KEY
If set, this bit indicates that this key is to be collated, using the collate file. This bit is set only if this key is unsigned.

KEY_LENGTH
Indicates the length of the key in bits.

KEY_DISPLACEMENT
Indicates the location of the key in bits from the beginning of the record. The first bit of the record is considered relative bit 0.

Delete Key Table

The Delete Key Table indicates which records to delete or include in the sort output file, depending on conditions indicated. The format of this table follows:

```
DECLARE 01 DELETE_KEYS (10)          BIT(48),
           02 KEY_TYPE               BIT(1),
           02 LOGIC_TYPE             BIT(1),
           02 DATA_TYPE              BIT(1),
           02 LENGTH_TYPE            BIT(1),
           02 COMPARE_TYPE           BIT(4),
           02 KEY_LENGTH             BIT(2),
           02 KEY_DISP_1             BIT(14),
           02 KEY_DATA               BIT(24),
              03 FILLER              BIT(10),
              03 KEY_DISP_2          BIT(14);
```

The meaning and possible values of the variables in the table follow:

KEY_TYPE
Indicates whether this key is an INCLUDE key or a DELETE key. 0 means INCLUDE, and 1 means DELETE.

LOGIC_TYPE
Indicates how this key is to be compared to the next key. 0 means OR, and 1 means AND.

DATA_TYPE
Indicates whether the key is to be compared to a literal or another field in the record. If 0, a literal is expected in KEYDATA, and if 1, a bit displacement from the beginning of the record is expected in KEY_DISP_2.

LENGTH_TYPE
Indicates whether this key is an ALPHA or NUMERIC key. This variable indicates whether the KEY_LENGTH refers to 8-bit bytes or 4-bit digits. 1 = byte, 0 = digit.

COMPARE_TYPE
Indicates how the field is to be compared. The possible values are as follows:

Value	Meaning
1	GTR
2	LSS
3	NEQ
4	EQL
5	GEQ
6	LEQ

KEY_LENGTH
Indicates the length of the key (refer to LENGTH_TYPE).

KEY_DISP_1
Indicates the location of the key in bits from the beginning of the record. The first bit of the record is considered relative bit 0.

KEY_DATA
If the comparison is to be made with a literal (refer to DATA-TYPE), this field contains the literal. The literal is right justified if shorter than 24 bits in length.

KEY_DISP_2
If the comparison is to be made with another location in the record, this field indicates the location of the field in bits from the beginning of the record. The first bit of the record is considered relative bit 0.

Merge Input Table

The Merge Input Table specifies the input files to be merged. The format of this table follows:

```
DECLARE 01 MERGE_INPUT_TABLE BIT(80),
          02 FILLER BIT(8),
          02 MERGE_DISK_IN BIT(8),
          02 MERGE_INPUT_FILE (8) BIT(8);
```

The meaning and possible values of the variables in the table follow:

MERGE_DISK_IN
Indicates how many of the input files are on disk.

MERGE_INPUT_FILE
Each element of the array, limited to the number of input files, specifies a relative file number of the calling program's file that is to be merged.

Sort Communicate

A program passes the sort communicate to the MCP when a sort is desired. The format of the sort communicate is as follows:

```
DECLARE 01 SORT_COMMUNICATE
           02 SORT_VERB                BIT(12),
           02 SORT_OBJECT              BIT(24),
           02 SORT_ADVERB              BIT(12),
             03 SORT_CT_RESTART        BIT(1),
             03 SORT_CT_DUPCHECK       BIT(1),
             03 SORT_CT_W1_PID         BIT(1),
             03 SORT_CT_W2_PID         BIT(1),
             03 SORT_MULTI_INPUT       BIT(1),
             03 FILLER                 BIT(7),
           02 SORT_CT1                 BIT(24),
           02 SORT_CT2                 BIT(24),
           02 SORT_CT3                 BIT(24),
           02 SORT_CT4                 BIT(24),
           02 SORT_CT5                 BIT(24),
           02 SORT_CT6                 BIT(24),
           02 SORT_CT7                 BIT(24),
           02 SORT_CT8                 BIT(24);
```

The meaning and possible values of the variables in the communicate follow:

SORT_VERB
This field contains the literal @01D@ to inform the MCP that this is a sort communicate.

SORT_OBJECT
This field contains the base-relative address of the Sort Information Table.

SORT_CT_RESTART
This field has the same function as SORT_RESTART in the Sort Information Table.

SORT_CT_DUPCHECK
This field has the same function as SORT_DUPCHECK in the Sort Information Table.

SORT_CT_W1_PID
Indicates that there is a work pack family name for work file 1 (refer to SORT_CT7).

SORT_CT_W2_PID
Indicates that there is a work pack family name for work file 2 (refer to SORT_CT8).

SORT_MULTI_INPUT
Indicates that the input file is contained on a multipack file.

SORT_CT1
This field contains the base-relative address of the Sort Key Table.

SORT_CT2
This field contains the calling program's file number of the input file to be sorted. If there is to be a merge, this field contains the base-relative address of the Merge Input Table.

SORT_CT3
This field contains the calling program's file number of the (sorted) output file.

SORT_CT4
This field normally contains @FFFFFF@. If collating is to be done, then this field contains the calling program file number of the collate file.

SORT_CT5
This field is always set to zero (0).

SORT_CT6
This field normally contains @FFFFFF@. If INCLUDE/DELETE keys are present, then this field contains the base-relative address of the Delete Key Table.

SORT_CT7
If work file 1 is to be assigned to a user pack, then this field contains a base-relative address pointing to a 10-character field, which contains the pack family name.

SORT_CT8
If work file 2 is to be assigned to a user pack, then this field contains a base-relative address pointing to a 10-character field, which contains the pack family name.

INDEX

ASCENDING 3-18
BIAS option 2-14
COBOL sorting 1-2
COLLATE option 2-14
comments 3-2
compiling sorts 2-1, 3-2
DATA specification 3-20
Delete Key Table A-7
DELETE specification 3-16
DISKSORT option 2-15
DUPCHECK option 2-16
file name 1-5
file naming conventions 1-5
FILE specification 2-2, 3-3
file title 1-5
fixed format specifications 3-22
FORCEDATA specification 3-21
FORCEKEY specification 3-19
free-form format specifications 3-2
header specification 3-23
INCLUDE option 2-17
INCLUDE specification 3-16
input file description 2-3, 3-3
input file description using file attributes 2-5, 3-5
KEY specification 3-18
listing 2-2
Merge Input Table A-8
no collate (NC) 2-12
option specifications 2-13, 3-10
output file description 2-7, 3-7
output file description using file attributes 2-9, 3-8
OVERFLOW specification 3-22
record specifications 3-16
REMAPS 3-19
RESTART option 2-20
right-signed numeric (RSN) 2-12
SEQUENCE option 2-21
Sort Communicate A-9
Sort Information Table A-2
Sort Key Table A-6
SORT specifications 2-2
SORT/MERGE 1-1, 2-2
SORT/QSORT 1-1
SORT/TAPESORT 1-1
SORT/VSORT 1-1, 2-15
STABLE option 2-21
SUMMARYDATA specification 3-21
syntax diagrams 1-2
SYNTAX option 2-21
TAGCOBOL option 2-21
TAGRPG option 2-22

INDEX (CONT)

TAGSEARCH option 2-22
TAGSORT option 2-22, 3-14
TAPESORT option 2-22
TEACH option 3-14
TIMING option 3-15
ZIP option 3-15
ZONE 3-19

Documentation Evaluation Form

Title: B 1000 Systems SORT Language Manual

Form No: 1152071

Date: August 1983

Burroughs Corporation is interested in receiving your comments and suggestions, regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

From:

Name _____

Title _____

Company _____

Address _____

_____

Phone Number _____ Date _____

Remove form and mail to:

Burroughs Corporation
Corporate Documentation – West
1300 John Reed Court
City of Industry, CA 91745
U.S.A.